SAND2017-3559R

IDC RE-ENGINEERING REPORT
SAND2017-XXXX R
April 2017

IDC Re-Engineering Phase 2 Data Model

Version 2.0

J. Mark Harris, Christopher Young, Benjamin Hamlet, Dorthe Carr, Hunter Knox

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia National Laboratories is a multi-mission laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Sandia National Laboratories

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

SAND2017-XXXX R
Unlimited Release
April 2017

IDC Re-Engineering Phase 2 Data Model

Version 2.0

J. Mark Harris, Christopher Young, Benjamin Hamlet, Dorthe Carr, Hunter Knox
Dynamic Monitoring Software
Sandia National Laboratories
P.O. Box 5800
Albuquerque, New Mexico 87185-MS0401

Abstract

This document describes a conceptual Data Model for use in the IDC Re-Engineering development project.

REVISIONS

Version	Date	Author/Team	Revision Description	Authorized by
1.0	1/15/2016	SNL IDC Re-Engineering Team	Release for E1	M. Harris
1.1	7/15/2016	SNL IDC Re-Engineering Team	Release for E2	M. Harris
1.2	1/15/2017	SNL IDC Re-Engineering Team	Release for E3	M. Harris
2.0	4/1/2017	SNL IDC Re-Engineering Team	Updated for E3 comments	M. Harris

CONTENTS

FIGURES

1 Introduction

1.1 Purpose

This document describes a conceptual data model to guide the development of a new seismic, hydroacoustic, and infrasound (SHI) sensor data processing system for the International Data Centre (IDC) (the System). As a conceptual data model, the data classes in this document are intentionally abstract and avoid describing implementation details required to realize the classes in particular programming languages or storage solutions. The intent of this data model is to identify important objects in the System, the information contained in each object, and the relationships between objects. It is intended to be used by domain experts and software engineers developing the System.

1.2 Scope

This data model is intended to address acquisition, processing, analysis, and distribution of data and products by the System.

Note that this current version is not complete; it is focused on core reference information, processing configuration, and processing results. The primary concern is identification of data classes and their important attributes, and relationships between these classes. Many additional classes and attributes are required for a complete data model.

1.3 Objectives

1.3.1 System Development

This data model is intended to replace prior data models that focused on data transport or data storage in a relational database with an object model applicable to object-oriented programming. Data transport and data storage are addressed by other efforts.

1.3.2 Data Provenance

An important feature of the System is the collection of information about how processing results were computed, referred to as data provenance. Data provenance is useful for fully understanding a processing result, given changes to the System or environment over time, and investigating the evolution of a result as the available information changes. It is often difficult or impossible to recreate this information if it is not captured when processing results are created. Data provenance is poorly captured in previous data models (e.g., CSS.3.0, SEED, QuakeML) and is a significant new aspect of this model.

This model addresses provenance in three ways:

1. Versioning of primary data objects such as data channels, signal detections, and events to capture the history of changes to those objects in the System.
2. Defining and capturing processing configuration and parameters and associating those values to a processing result.

3. Capturing creation information (creation source and time) on all objects to allow connection to general System configuration and other information.

1.4 Conventions

This report contains diagrams and descriptions for different parts of the data model. This section explains the notation used for these diagrams.

1.4.1 Classes and Associations

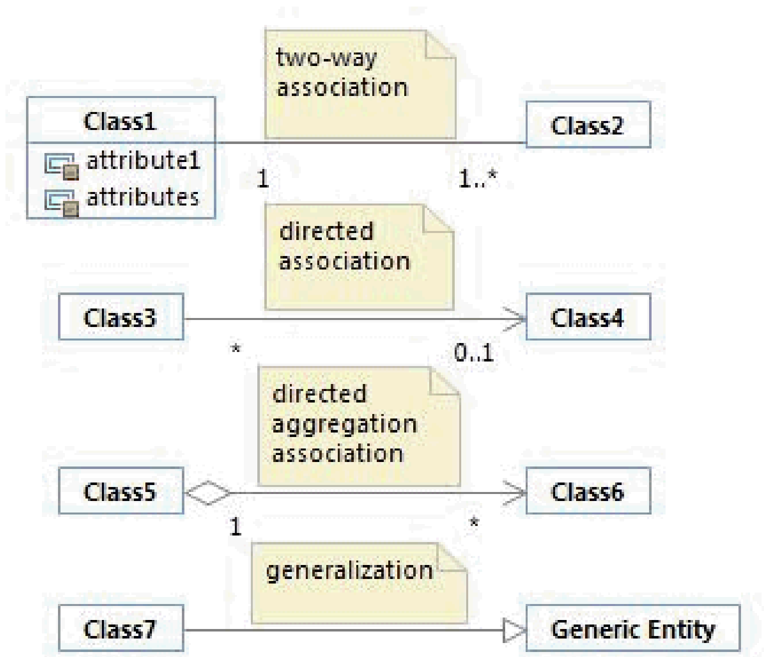

Figure 1. Class Notation

Figure 1 illustrates class and association notation. Each of the boxes (e.g., the box named 'Class1') represents a class. Classes have attributes and associations to other classes.

Each named item in a class is an attribute of that class. These fields can represent primitive values, collections, or instances of other data model classes. The conceptual data model is primarily concerned with identifying important classes and their relationships, so details about specific data types or representations of attributes are often omitted. These details will be elaborated, as the conceptual data model is refined into logical or physical models, prior to implementations in specific programming languages and databases.

Relationships between classes are shown as lines and can be modeled in different ways. The line from Class1 to Class2 is a two-way association. Association relationships represent loose coupling between classes and should be thought of as an instance of one class having a reference to an instance of the associated class. This means that instances of Class 1 have references to instances of Class 2 and that instances of Class 2 have references back to those instances of Class 1. Each end of a relationship is marked with a number indicating the number of instances of the class that can participate in the relationship. This is known as the relationship's multiplicity (or cardinality). The '1..*' on the right means that instances of Class1 have references to one or more instances of Class2, and the "1" on the left means instances of Class2 have references to exactly one instance of Class1.

The arrow from Class3 to Class4 is a directed association. This means instances of Class3 have references to instances of Class4, but instances of Class4 do not have references back to instances of Class3. The multiplicity rules work the same way as described above: Class3 has a reference to zero or one Class4, but a Class4 can be referenced by any number of Class3s.

The arrow with a diamond at the tail from Class5 to Class6 is a directed aggregation association. While the previous associations were only references to other classes, a directed aggregation association defines ownership over the lifecycle of the associated classes. Here, a Class5 aggregates Class6s. This means Class5 controls the lifecycle of all Class6s it aggregates, and that no Class6 can exist without its parent Class5. The multiplicity rules are the same as explained above. Class5 aggregates zero or more Class6s, and a Class6 must be aggregated by exactly one Class5. Though not modeled, it is assumed there is a way to traverse directed aggregation associations backwards. In this diagram, this means that given an instance of Class6, there is a way to determine which instance of Class5 aggregates that Class6 instance.

The arrow with a triangular arrowhead from Class7 to the class Generic Entity is a generalization/specialization relationship. This means that Class7 inherits attributes and methods from Generic Entity (i.e., Generic Entity is a base class and Class7 is a specialization of that base class). Class7 may also be referred to as a specific "type of" a Generic Class.

1.4.2 Enumerations

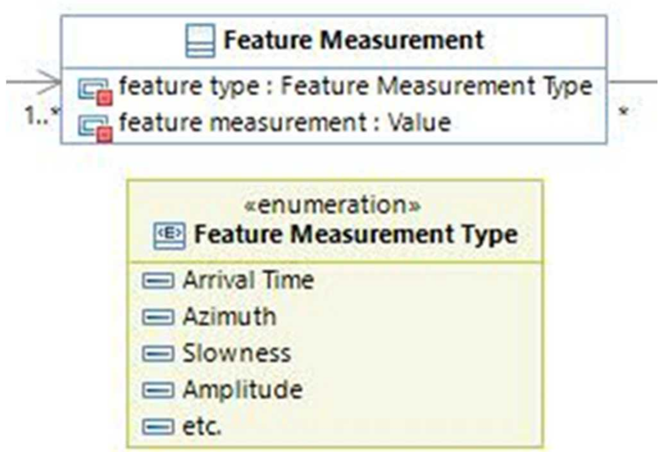

Figure 2. Enumeration Notation

Enumerations are used in the data model when particular values are expected for a class attribute. Attribute types may be specified in a class using the format "attribute name :

attribute type”. The attribute type may identify an enumeration as the data type. The example in Figure 2 shows the class **Feature Measurement** with an attribute **feature type** that has data type “Feature Measurement Type”. The values expected for **feature type** are listed in the Feature Measurement Type enumeration box.

1.4.3 Constraints

In some cases a constraint on the use of the data model is intended but not modeled in UML (e.g., a particular type of measurement must be associated for an object to be valid). These constraints are specified in notes on the diagram or the text description.

1.4.4 Text

In the class descriptions below, class names and attributes are **bold** text. Note that some terms (for example, calibration) are used both generically and as a class name. When referring to the class, the term should be bold.

2 Class Diagrams

2.1 Overview

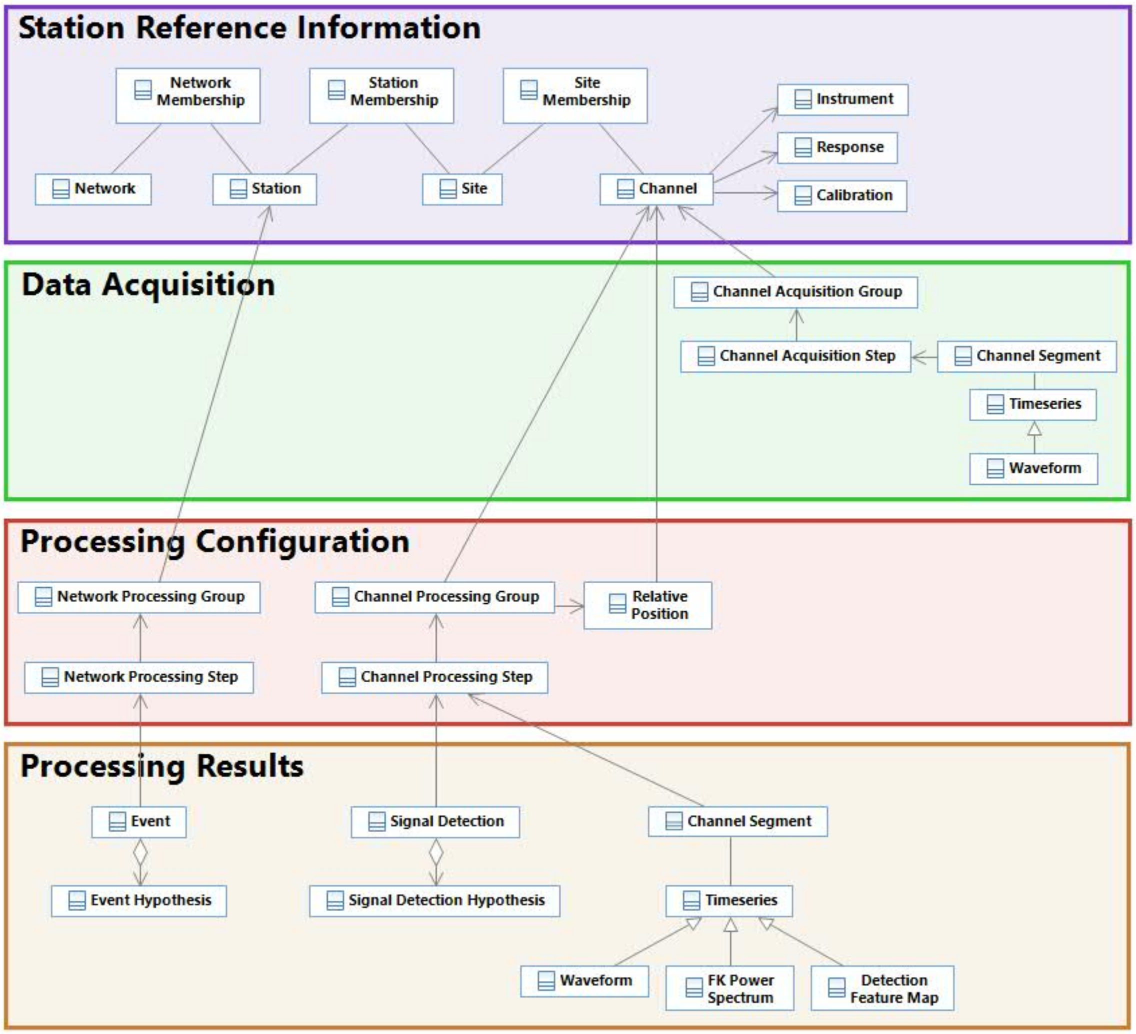

Figure 3. Organization of Data Model

The overall organization of the data model shown in Figure 3 attempts to separate information of different types and lifecycles. A goal of this data model is to maintain a clear distinction between reference information, data processing configuration, and processing results. The current data model identifies four categories of data (other categories will be needed as the data model expands):

1. Station Reference Information
2. Data Acquisition
3. Processing Configuration
4. Processing Results

Station Reference Information classes provide relatively static information about the sources of sensor data. This information is traditionally organized into a hierarchy of membership (**Networks**, **Stations**, **Sites**, and **Channels**) based on the physical and administrative structure of stations in the field. It is often provided by the station operators as authoritative information about the station and its sensors. Station Reference Information may be considered internally complete and independent from the System. The System treats this as static configuration information.

Data Acquisition classes provide configuration of data acquisition components and capture of raw sensor data. Once acquisition for a core network of sensors has been configured, this information is relatively static, typically changing only when new data sources are introduced. Data acquisition may be a separable, independent function of the System.

Processing Configuration classes cover the static configuration of processing sequences and algorithms used by automated and interactive processing. A System Maintainer sets this configuration when the System is deployed, and it is rarely changed on the System as it is running.

Processing Results include the mission data objects that are computed by the System; for example, events, signal detections, and processed waveforms. This information grows as the System processes more data. Processing results are associated with the processing configuration objects, the specific parameters, and other processing results that were used to create them. A fundamental expectation of the System is that previous results are not overwritten as new results are obtained (i.e., multiple versions of processing information are retained to capture the history of changes to that result).

2.2 *General Provenance*

2.2.1 Configuration Info

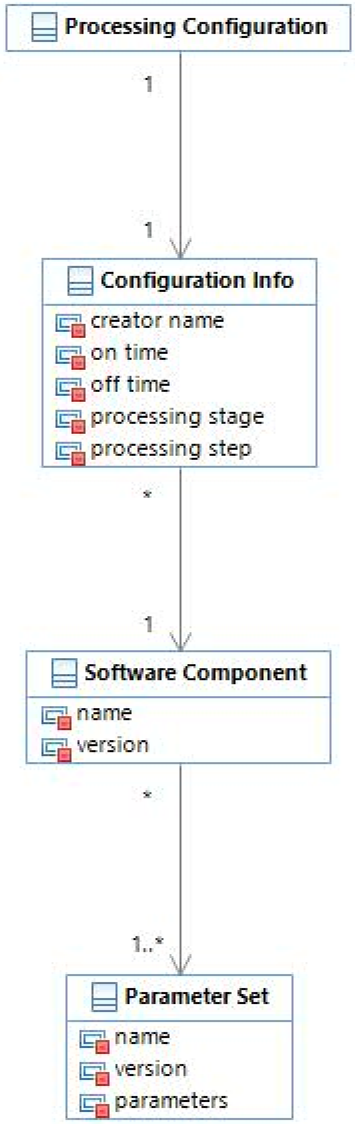

Figure 4. Configuration Info Classes

The **Configuration Info** classes (Figure 4) capture the processing parameters for all data acquisition and processing operations of the System. **Configuration Info** specifies when the particular processing operation will occur, what the processing will be, and how it will be done.

The **Configuration Info** class includes the time span during which the processing operation is valid (**on time**, **off time** – not all processing operations will be configured to run indefinitely), the **processing stage** and **processing step** where the processing operation is intended to run (e.g., pre-analyst review automatic processing, body wave magnitude estimation), and the name of the person/entity that configured the processing operation (**creator name**).

A data processing operation always involves a **Software Component**. The **Software Component** class includes a **version** attribute to track changes to the software components over time. The parameters to be used by a **Software Component** are captured in the associated **Parameter Set** class.

2.2.2 Creation Info

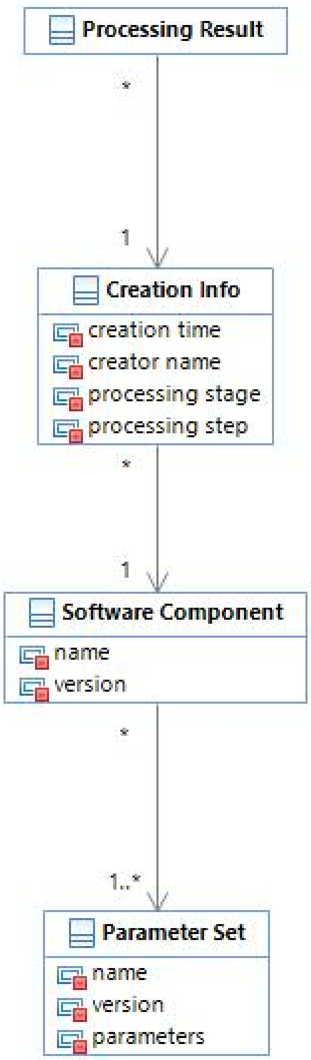

Figure 5. Creation Info Classes

The System captures **Creation Info** (Figure 5) for all data acquisition and processing results (e.g., events, signal detections, processed waveforms, etc.). The **Creation Info** class is not shown on all diagrams in this document, but it is assumed to be associated with all processing result classes.

Creation Info provides important information if a user wants to understand when, why, and how the System produced a certain result. **Creation Info** captures the provenance of the data processing done by the System.

The **Creation Info** class includes the time when the product was created (**creation time**), the **processing stage** and **processing step** where the product was created (e.g., pre-analyst review automatic processing, body wave magnitude estimation), and the name of the person/entity that created the product (**creator name** – e.g., short-term average/long-term average (STA/LTA) seismic signal detection, analyst John Smith).

The creation of a data processing result always involves a **Software Component** (even for manually created products, such as analyst-added signal detections). The parameters used by a **Software Component** for a particular processing result are captured in the **Parameter Set** class.

Note that there is a lot of similarity between the Configuration Info and Creation Info classes. This is intentional. For a processing result created by a processing operation linked to Configuration Info, the Creation Info would be expected to be the same, suggesting that Creation Info should directly reference Configuration Info and only need add a few attributes such as **creation time**. However, in some cases the System or the user may choose to over-ride some of the Configuration Info, so Creation Info must capture the actual information used, thus insuring accurate data provenance.

2.2.3 Value

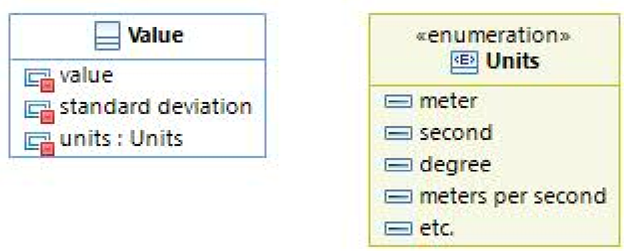

Figure 6. Value Classes

Every numeric measurement (e.g. a signal detection's onset time) consists of a measured **value**, the **standard deviation** (characterizing the uncertainty), and the **units** of measurement, collected in the **Value** class (Figure 6). Other classes contain attributes with type of **Value**. For example, the **Feature Measurement** class (see Section 2.6.3) combines a **Value**, called **feature measurement**, with a **feature type** attribute describing what was measured.

The **Value** class can also represent numeric predictions (e.g., a predicted signal detection arrival time) since these predictions are expectations of values that could be measured.

2.3 *Station Reference Information*

2.3.1 Network, Station, Site, and Channel

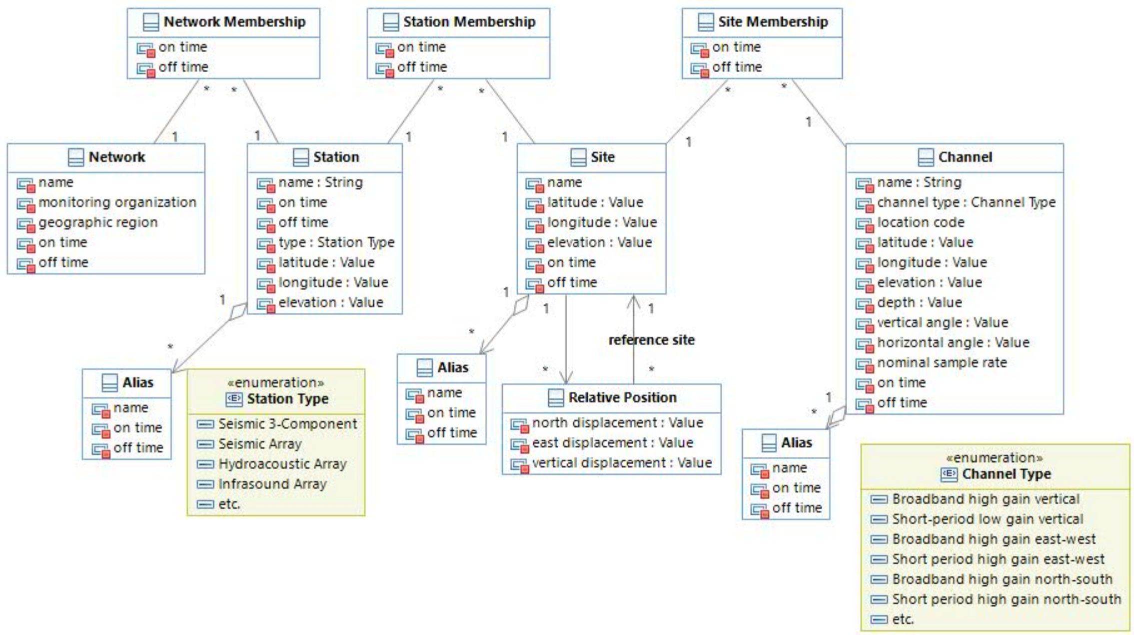

Figure 7. Network, Station, Site, and Channel Classes

Figure 7 illustrates the primary classes and relationships for Station Reference Information, including the **Network**, **Station**, **Site**, and **Channel** classes.

A **Network** is a collection of affiliated **Stations.** Often, the relationship is that these **Stations** are operated by a particular **monitoring organization**, but a **Network** can be arbitrarily defined as well (e.g., a set of **Stations** that a researcher is using for a project). A **Network** has a **name** (e.g., "GDSN"), and an **on time** and **off time**. The set of affiliated Stations is captured in the **Network Membership** class, with **on time** and **off time** detailing when a **Station** was associated with a **Network**. The same **Station** can belong to more than one **Network**.

A **Station** has a **name**, as well as an **on time** and an **off time**, indicating the time range the **Station** was active. It also has a **station type** (e.g., seismic three-component, hydroacoustic array). A **Station** has a location (**latitude**, **longitude**, **elevation**, **depth**). This information can be used for indicating the general location of the **Station** (e.g., on a map), as opposed to the detailed position of individual sensors (described subsequently). **Stations** may be known by two or more different names (e.g., ASAR and PS3). Because of this, in addition to its primary **name**, a **Station** may have a set of **Aliases**.

A **Station** is a collection of affiliated **Sites** as recorded by the **Station Membership** class. A **Station Membership** records when a **Site** was associated with a **Station** and who created the membership. A **Station** can have zero or more **Sites**. For example, an array is a type of **Station** consisting of three or more **Sites,** arranged to enhance signal coherency in a particular frequency band.

A **Site** is a physical installation (e.g., building, underground vault, borehole), potentially containing a collection of sensors**.** A **Site** has a location (**latitude**, **longitude**, **elevation**), a **name** (e.g., “MK01”), the time it was activated (**on time**), and the time it was deactivated (**off time**). As a **Site** may be known by two or more names, a **Site** may also have a set of **Aliases**. Historically, for the various **Sites** within an array, **Relative Position** was specified relative to some reference **Site**. The data model includes this relationship to accommodate historical information.

A **Site** includes a collection of affiliated **Channels,** as recorded in the **Site Membership** class. A **Site Membership** records when a **Channel** was associated with a **Site** and who created the membership. A **Site** can have zero or more **Channels**. For example, for a typical three-component **Station**, there is one **Site** with three broadband **Channels,** measuring vertical, north-south, and east-west ground motion.

A **Channel** is an identifier for a data stream from a sensor measuring a particular aspect of some physical phenomenon (e.g., ground motion or air pressure). A **Channel** has metadata, such as a **name** (e.g., “BHZ” is broadband ground motion in the vertical direction), **on time** and **off time**, and a channel **type** that encodes the type of data recorded by that sensor. There are different conventions for **Channel** naming, so a **Channel** can have **Aliases**. The **Channel** class also includes information about how the sensor was placed and oriented: **depth** (relative to the **elevation** of the associated **Site**), **horizontal angle**, and **vertical angle**.

A **Site** may have one or more redundant sensors measuring the same physical phenomenon, each with their own **Channel**. For example, a **Site** could have two **Channels** with the same **name** (e.g., “BHZ”), each corresponding to primary and backup sensors of the same type. These **Channels** can be differentiated by their **location code** (e.g., 0, 1, 2).

2.3.2 Instrument Response

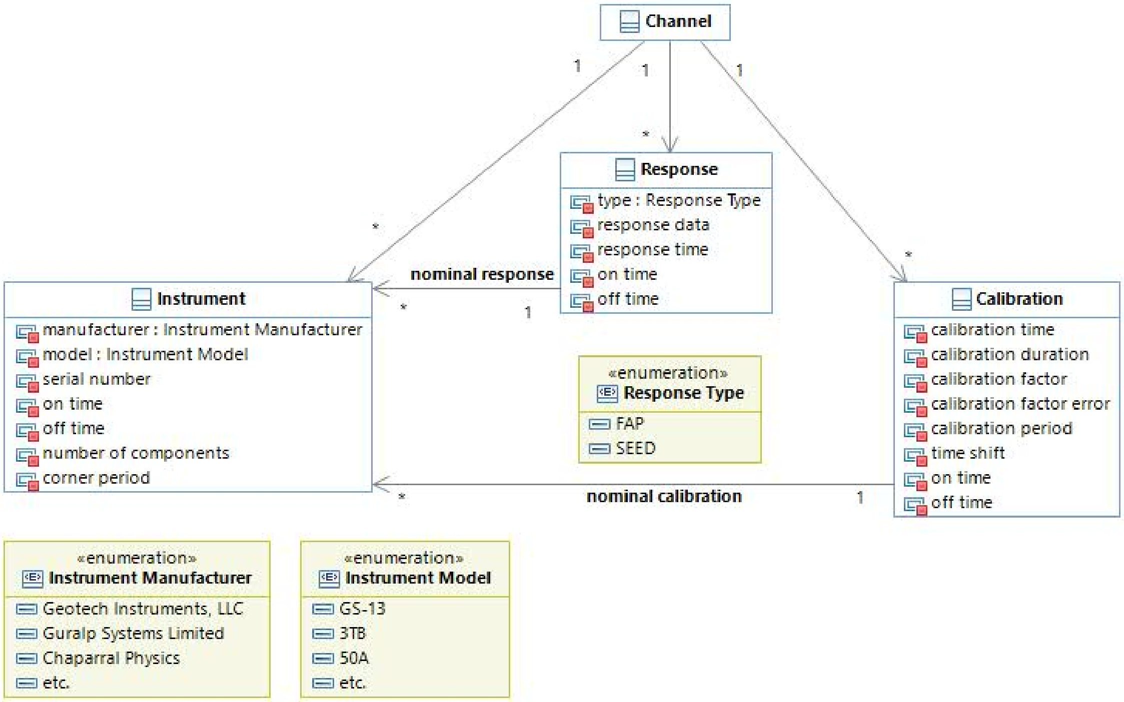

Figure 8. Response Classes

As described above, a **Channel** is an abstract entity specifying a basic type of measurement capability; the actual hardware that produces the data stream for that channel is an instrument (Figure 8). The **Instrument** class has a **manufacturer**, a **model**, a **serial number**, and nominal calibration and response information (i.e., the specifications provided by the manufacturer), which are captured in the **Calibration** and **Response** classes. While the type of information that a **Channel** records will not change, the actual instrument used may (e.g., an upgrade to a more current model); hence **Channel** can point to more than one **Instrument**, and **Instrument** includes **on time** and **off time** as attributes.

Periodically, the instrument corresponding to a **Channel** is calibrated to characterize the true relationship between the underlying phenomenon the instrument is measuring and the actual output of the instrument. As with the manufacturer-provided calibration information, this calibration information is stored in the **Calibration** and **Response** classes. Calibration information is used to convert the output of the instrument (e.g., volts, counts) into the phenomenon that the instrument is measuring (e.g., seismic ground displacement). **Calibration** includes information about when an instrument was calibrated and what the response was for a particular period (i.e., inverse of frequency). **Response** includes the full response function across a range of frequencies. The **calibration time** and **response time** attributes capture when the calibration was actually performed in **Calibration** and **Response** respectively, and both classes also include **on time** and **off time** as attributes, in order to track when the response information was available for use by the System.

A three-component sensor (i.e. three sensors physically embedded in a single unit) can be described in this class with three separate **Channel** objects pointing to a single **Instrument**. Furthermore, assuming that the aforementioned three-component sensor has the same nominal response and calibration information, these three **Channel** objects would be associated with a single nominal **Calibration** and **Response**, each of which references the **Instrument**. However this model also allows for separate nominal information per channel. The information found in the **Instrument** class (e.g. the number of components and the corner frequency), coupled with the nominal sample rate (found in the **Channel** class), provide the details for determining the **channel name** according to SEED convention.

2.4 *Data Acquisition*

2.4.1 Channel Acquisition Processing

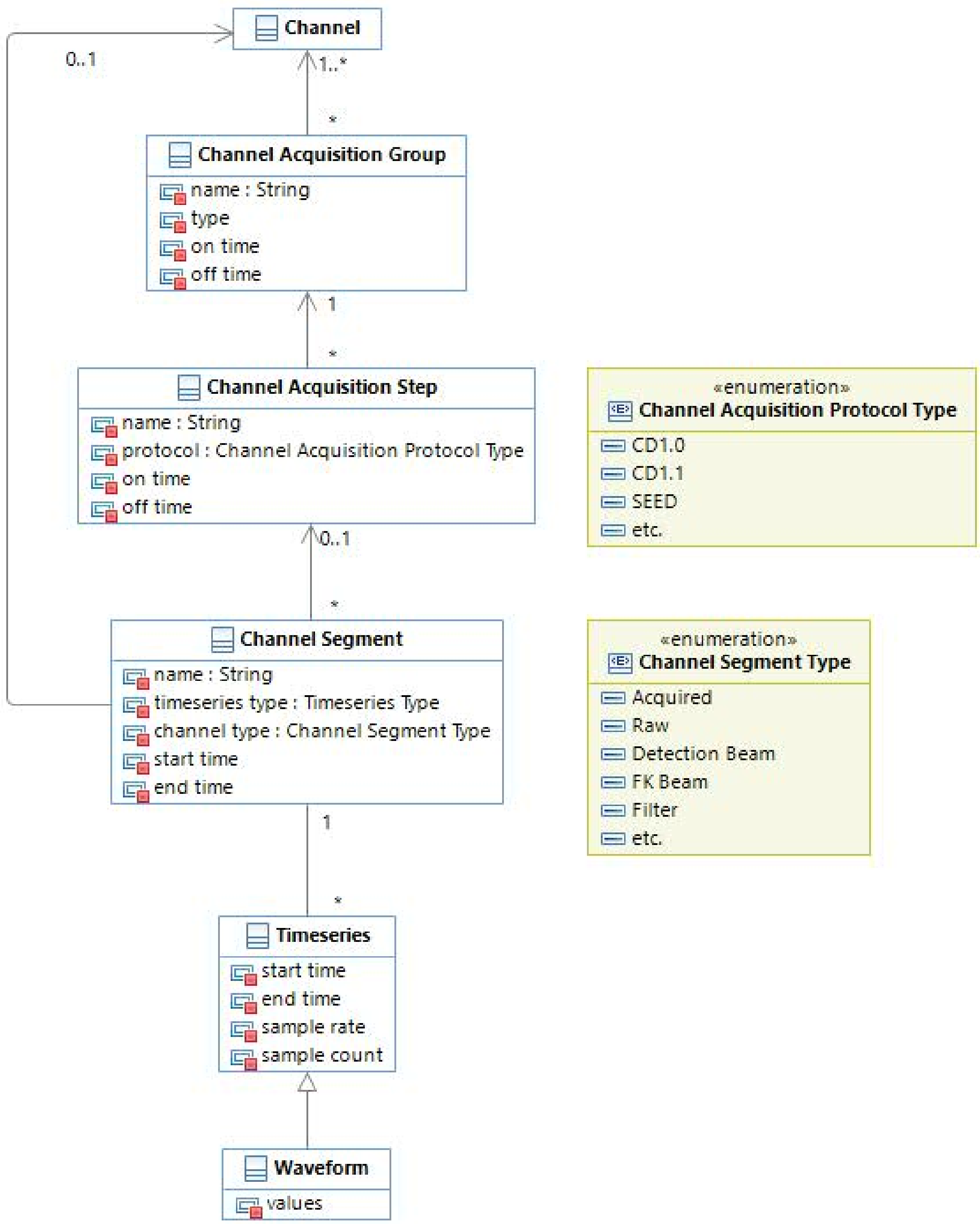

Figure 9. Channel Acquisition Processing Classes

Acquisition of data by the System is represented by classes that specify the configuration of the acquisition software, and storage of the data as **Channel Segments** and **Waveforms** (Figure 9).

A **Channel Acquisition Step** is defined for each instance of the acquisition software, specifying the protocol used by the Station or other data source (CD1.0, CD1.1, SEED, etc.). The **Channels** to be received by that instance are defined by the **Channel Acquisition Group**, which includes one or more **Channels**. This configuration should match the actual sensor data channels transmitted as a group by the station. A station may send one or more data streams, grouping its channel data as appropriate to the station.

Raw acquired **Channel** data are represented by a sequence of **Channel Segment** objects. These **Channel Segments** refer to the **Channel Acquisition Step** that received the data and provide basic information about the data (**start time**, **end time**, **sample rate**, etc.), but do not include the data itself. **Channel Segment** also has an attribute **channel segment type**, to indicate the type of data in the segment. For raw acquired data this type is always set to “acquired”. The data itself is contained in **Waveform** objects, which are specializations of the **Timeseries** base class. Note that each **Channel Segment** points to one or more **Timeseries**. The relationship is one to many to allow for the possibility that a long **Channel Segment** may have gaps within it, hence the actual data can be broken into a set of **Waveforms** with valid data, rather than requiring one longer **Waveform** with the gaps filled.

2.5 Processing Configuration

The System is envisioned to include a capability to automatically execute multiple processing sequences. A processing sequence is a standardized way to specify the processing steps in the sequence and the flow of control between the steps. Control flow may include sequential, parallel, branching, and merging connections. Configuration of each step includes the triggering criteria for that step (for example, the minimum length of waveform data that is required to run a signal processing operation), as well as any algorithm configuration specific to this step. Processing sequence steps may include single-channel and multiple-channel signal processing operations, array processing operations, signal detection algorithms, station processing operations, and network processing operations, such as event formation.

The Examples section of this document includes some simple examples of processing sequence configuration and results.

2.5.1 Channel Processing

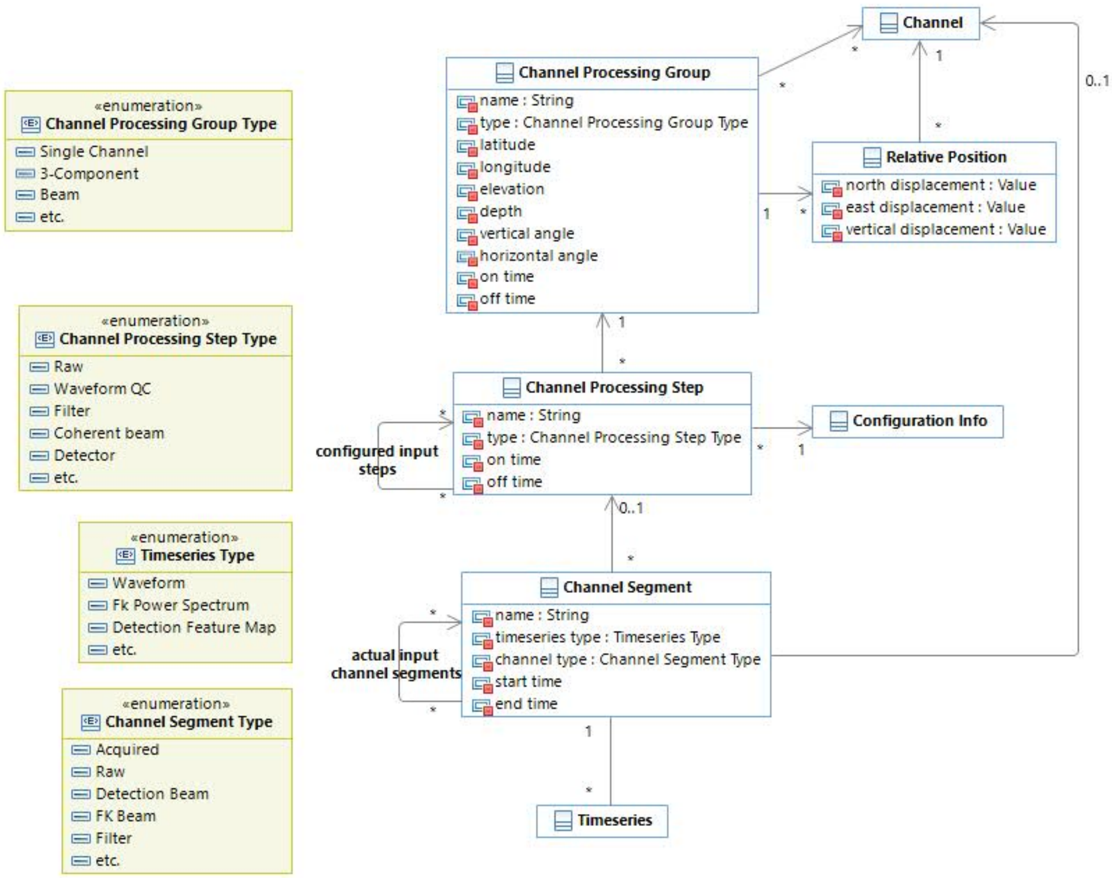

Figure 10. Channel Processing Classes

The system processes waveform data from **Channels** to produce processed **Channel Segments** and ultimately **Signal Detections** and **Feature Measurements**. Each of these processing operations is identified as a **Channel Processing Step** (see Figure 10). These steps may be grouped into a processing sequence by setting their **configured input steps**. Each **Channel Processing Step** references a **Channel Processing Group** that identifies the set of channels that it requires (a step may use only a subset of these channels). The purpose of the processing step is captured in the **Channel Processing Step type** attribute.

Each **Channel Processing Step** references a **Configuration Info** object for its specific configuration values. If the **Channel Processing Step** results in **Channel Segments** and corresponding **TimeSeries**, then these relationships are captured as shown. Note that the actual input data that were used may be different than was configured in the **Channel Processing Step** (e.g., perhaps one element of an array is noisy and thus is excluded from a beam), hence the **Channel Segment** that is produced references the actual input **Channel Segments** that were used. Note that processing steps could also result in **Signal Detection Hypotheses** and/or **Feature Measurements**, but these relationships are not shown here for brevity.

To support multi-channel processing operations at a **Site** or **Station**, the **Channel Processing Group** includes attributes to specify the location and orientation of the processing result (for example, the location of the "beam point" of an array, or the orientation of a rotated signal at a three-component sensor). For array processing, the relative position between the **Channels** may be specified using the **Relative Position** class. This could be desirable for efficiency (so relative positions don't have to be constantly recalculated whenever they are needed for a processing operation), or to support more accurate information, if it is available for the **Station**. Note that for multi-channel processing operations, the reference from the **Channel Segment** back to **Channel** is null (i.e. the relationship is only intended to reference a single raw channel).

2.5.2 Network Processing

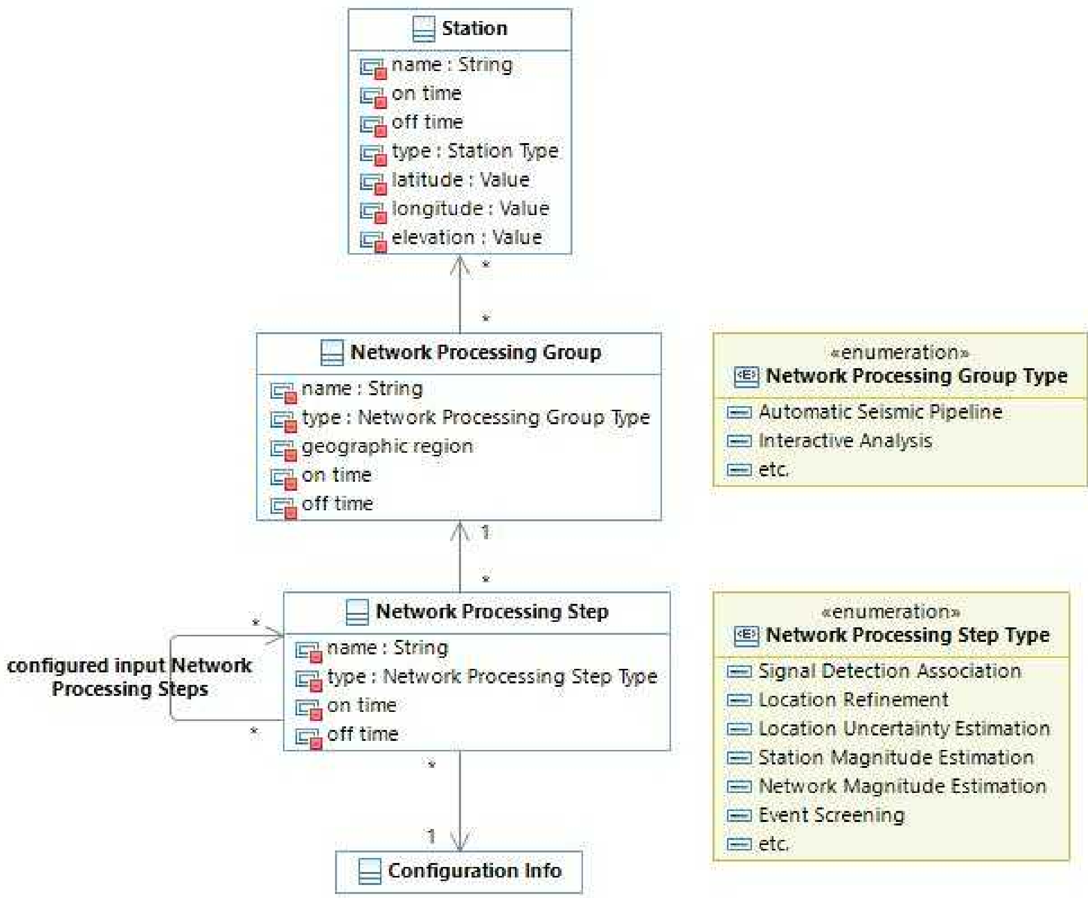

Figure 11. Network Processing Classes

Network processing operations are configured to work with data from a set of stations (for example, associating signal detections to form events). A network processing operation is identified as a **Network Processing Step** (see Figure 11). These steps may be grouped into a processing sequence by setting their **configured input Network Processing steps**. Each **Network Processing Step** references a **Network Processing Group** that identifies a set of **Stations** to be processed. The **Network Processing Step** references a **Configuration Info** object for its specific configuration values. An example of a sequence of network processing steps is shown in Figure 26.

2.6 Processing Results

2.6.1 Channel Segment

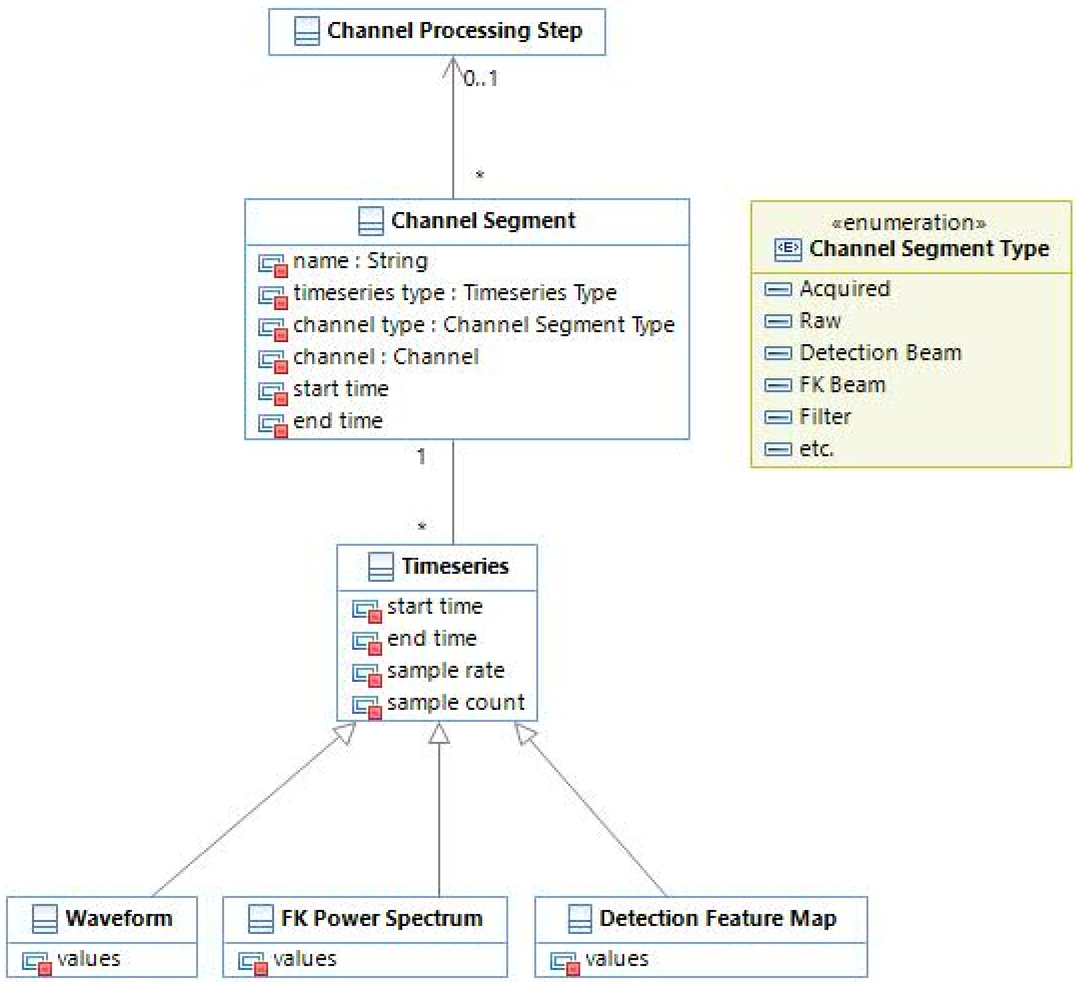

Figure 12. Channel Segment Classes

Figure 12 shows the **Channel Segment** classes. The **Channel Segment** class is the highest-level class representing time series data in the System. A **Channel Segment** contains general attributes about the time series, such as **channel type** and the **start time** and **end time** of the data segment, but does not include the data itself. For convenience, it also has a reference to the **Channel** that was the original source of the data (this relationship is empty in the case of multiple channel inputs). A **Channel Segment** refers to the **Channel Processing Step** that produced it (or the **Channel Acquisition Step** for raw data, as shown in Figure 9).

Each **Channel Segment** may be associated with a set of **Timeseries** objects (the actual data). The association of **Channel Segment** to **Timeseries** is optional, in the case that the **Timeseries** data was not persisted by the System, or when it may be easily recomputed from information in the **Creation Info** object and the referenced **Channel Processing Step**. This association is important for data with dropouts/gaps, as discussed previously.

Three example types of **Timeseries** are shown here: **Waveform**, **FK Power Spectrum**, and **Detection Feature Map**. Each of these has information they inherit from the base class (**start time**, **end time**, **sample rate**, **sample count**), but beyond that, the details vary:

- A **Waveform** is a one-dimensional **Timeseries** representing a geophysical measurement collected from a **Channel**, or a derived quantity created by a **Channel Processing Step** (e.g. a beam).

- An **FK Power Spectrum** is a multi-dimensional collection of power values derived from **Waveform** data from a collection of nearby sensors. The associated **Channel Processing Step** is an fk (frequency-wavenumber) transform, which converts a collection of **Waveforms** from three or more **Sites** into a time-by-slowness (north-south and east-west) array of coherency values, over a given two-dimensional range of slowness values, and filtered over a given frequency band. This is the primary way that array stations can accurately measure the azimuth and slowness of an arrival of energy.

- A **Detection Feature Map** is a generalized matrix of values for a particular feature over time as produced by a **Channel Processing Step** associated with a collection of channels from a station. The matrix contains a feature vector calculated for each point in time based on the processing of one or more channels from the station. The feature vector is a set of values indexed by secondary independent variables (e.g., frequency). For example, the Progressive Multi-Channel Correlation (PMCC) processing algorithm used for infrasound signal detection produces a **Detection Feature Map**, consisting of coherency, azimuth, and apparent velocity as a function of time and frequency.

2.6.2 QC Mask

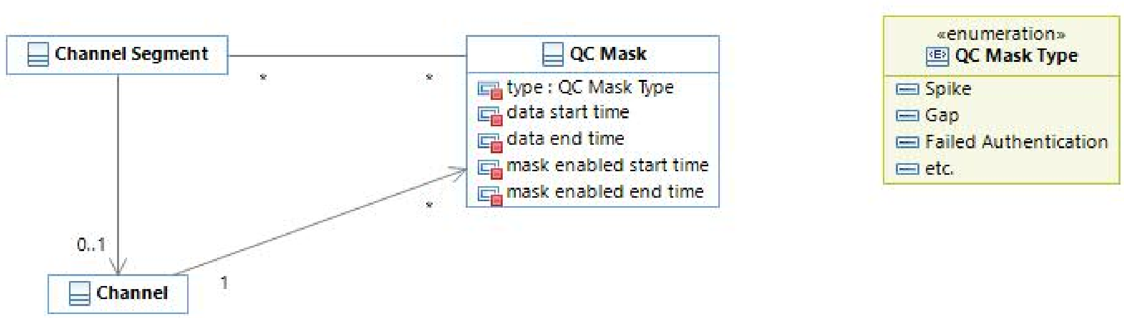

Figure 13. QC Mask Classes

A **QC Mask** marks an interval of channel data where there are data quality issues (Figure 13). A **QC Mask** may be used by a channel processing operation to determine if the data are of sufficient quality to be included in the operation. The use of **QC Masks** depends on the algorithm and policy of the monitoring organization. The **QC Mask** class includes the type of mask, as well as two different types of timing information. The **data start time** and **data end time** refer to the time in the data where the problem was identified, whereas the **mask enabled start time** and **mask enabled end time** attributes describe when the mask was available for use by the System.

A **Channel Segment** may have multiple intervals with problems (or multiple types of problems for the same interval), hence it can be associated with multiple **QC Masks**. Conversely, a given **QC mask** may apply to data from more than one Channel (e.g. when all the channels of an array drop out), so a **QC Mask** object may be associated with multiple **Channel Segment** objects.

2.6.3 Signal Detection and Feature Measurement

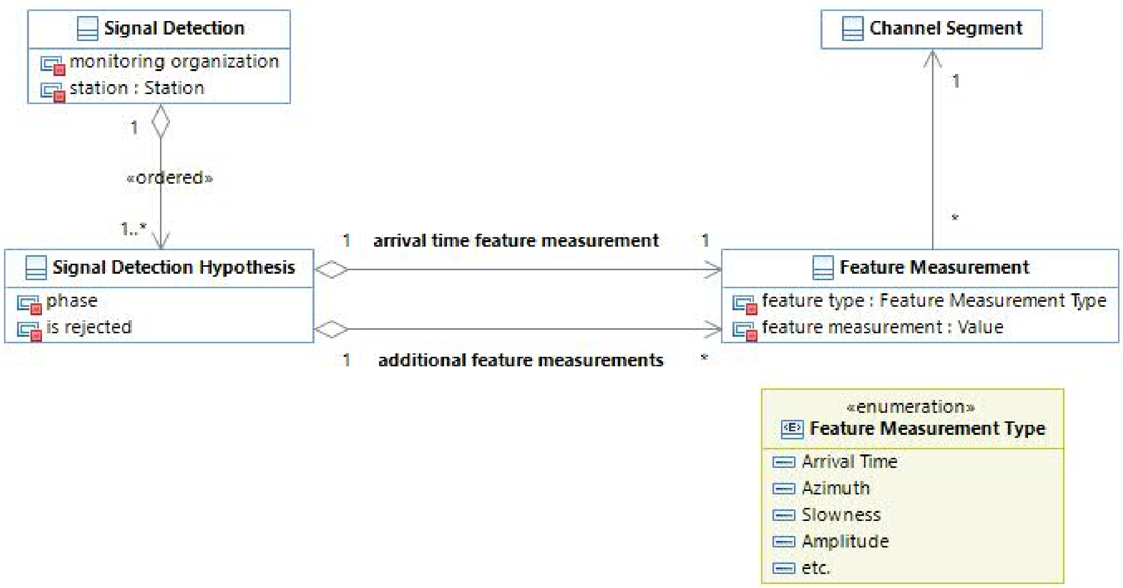

Figure 14. Signal Detection and Feature Measurement Classes

A **Signal Detection** represents the recording of the arrival of energy at a station (Figure 14). Determining a consistent solution for a **Signal Detection** (e.g., its phase identification and arrival time) is often an iterative process. A **Signal Detection Hypothesis** represents a proposed explanation for a **Signal Detection** (e.g., it is a P phase at a particular time). A **Signal Detection** can have multiple **Signal Detection Hypotheses**: for example, a computer algorithm could create the original **Signal Detection Hypothesis**, assigning a phase and an arrival time; an analyst reviewing that hypothesis may then choose to change the phase and/or the arrival time, hence creating a new **Signal Detection Hypothesis**. **Signal Detection** has an ordered list of its **Signal Detection Hypothesis** tracking how the **Signal Detection** was updated over time. **Signal Detection Hypothesis** also includes an attribute to track **Signal Detection Hypotheses** that were rejected during a particular processing stage (**is rejected**), in order to prevent their re-creation in subsequent processing stages. Note that processing stage is tracked through the **Creation Info** class attached to **Signal Detection Hypothesis**.

A **Signal Detection Hypothesis** typically will have many measurements associated with it, captured with the **Feature Measurement** class. **Feature Measurement** has been made generic to accommodate any new types of measurement that may be added in the future. Each **Feature Measurement** has a type indicated with the **feature measurement type** attribute, a **value**, and a reference to the **Channel Segment** on which it was calculated. As shown in the association above, each **Signal Detection Hypothesis** is required to have at least an arrival time **Feature Measurement**. The additional **Feature Measurements** are a "zero to many" relationship, because they are not required by the system.

2.6.4 Feature Prediction

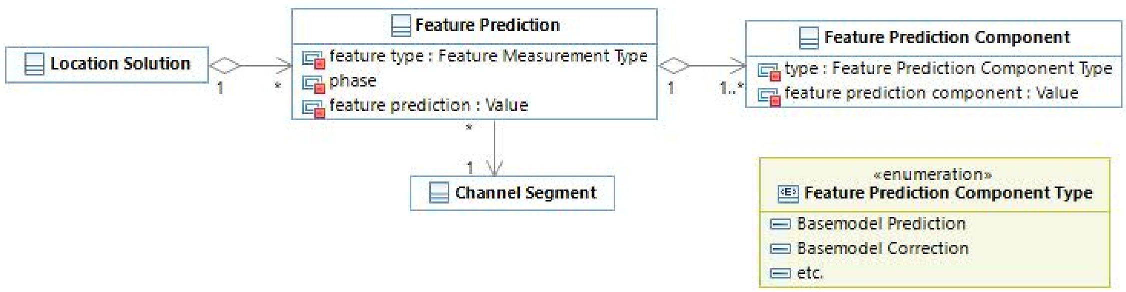

Figure 15. Feature Prediction Classes

A **Feature Prediction** captures a predicted value for a particular type of measurement (e.g., an arrival time, azimuth, slowness, or amplitude for a signal detection), calculated using an Earth model or empirical information (Figure 15). A **Feature Prediction** has a **feature type**, a **phase**, and a **feature prediction** (value). It contains a set of **Feature Prediction Components** and their associated values. For example, if the **feature type** is arrival time, then the **Feature Prediction** might consist of several **Feature Prediction Components:** a basemodel prediction from a global Earth model such as AK135, a basemodel correction that accounts for the elevation of the station above sea level, and a basemodel correction that accounts for the elliptical shape of the Earth.

2.6.5 Event

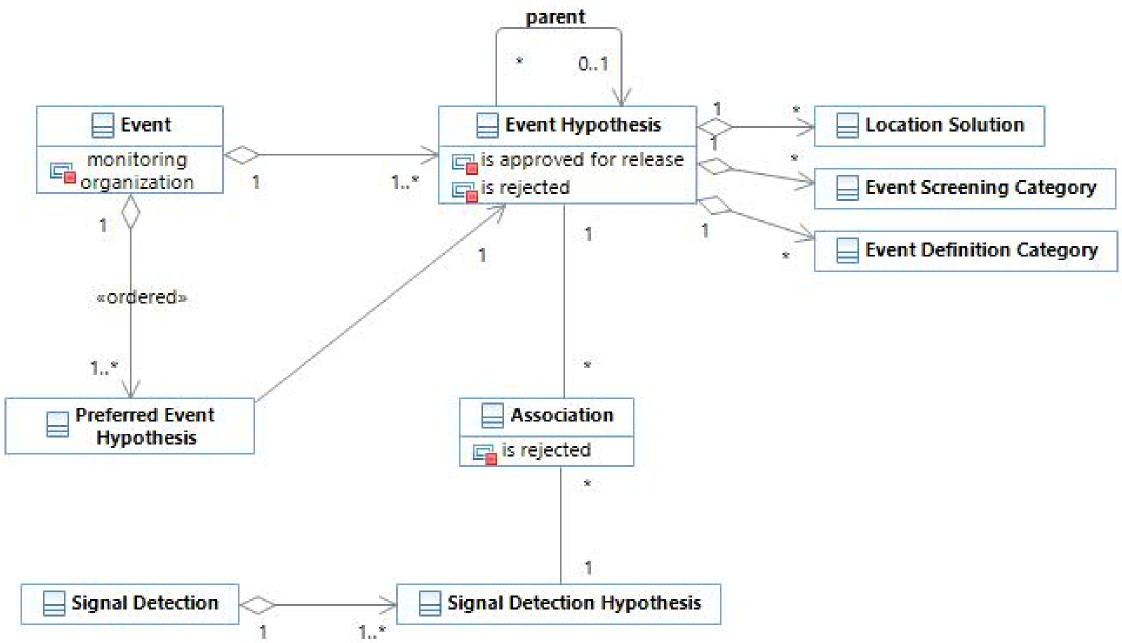

Figure 16. Event Classes

An **Event** marks the occurrence of some transient source of energy (e.g. a nuclear test) in the ground, oceans, or atmosphere (Figure 16). Because the System ingests **Events** from other systems (e.g., the International Seismological Centre (ISC), the National Earthquake Information Center (NEIC)), an **Event** has as an attribute specifying the **monitoring organization** that created it. **Events** created by different monitoring organizations are not combined, to ensure that System results are separated from external results.

Determining the parameters for an **Event** is often an iterative process for a monitoring organization. The **Event Hypothesis** represents a proposed explanation for an **Event**, such that the set of **Event Hypotheses** grouped by an **Event** represents the history of that **Event** (e.g., automatic computer processing might generate an initial **Event Hypothesis**, while subsequent refinement by an analyst might result in a different **Event Hypothesis**). A processing stage can have multiple **Event Hypotheses**, but only one **Event Hypothesis** can be designated as the **Preferred Event Hypothesis** for each processing stage. The history of the **Preferred Event Hypothesis** is preserved as an ordered list so that its evolution is available. The **rejected** attribute of an **Event Hypothesis** for a given processing stage is used to ensure that any rejected **Event Hypothesis** will not be rebuilt in subsequent processing stages. Only one **Event Hypothesis** can be designated as the overall **Preferred Event Hypothesis** for the **Event**, across all processing stages.

An **Event Hypothesis** is based on a set of associated **Signal Detection Hypotheses**. Choosing which set of **Signal Detection Hypotheses** is associated with an **Event Hypothesis**, and what phases each of those Hypotheses represent, is what is known as "building" an **Event**, and can be accomplished either automatically by an event-building algorithm, or manually by a human analyst. An **Association** represents this linkage between **Event Hypotheses** and **Signal Detection Hypotheses**. The **rejected** attribute of an **Association** is used to ensure that any rejected **Associations** will not be re-formed in subsequent processing stages. During analysis, a **Signal Detection Hypothesis** can be associated with multiple **Event Hypotheses**, but must be associated with only one preferred **Event Hypothesis** for that stage before the processing stage is completed.

Each **Event Hypothesis** has one or more **Location Solutions**, as described in the next section. Each **Event Hypothesis** may have an **Event Definition Criterion**, which is based on the **Event Definition Characteristics** (see Section 2.6.9), and ultimately determines if the **Event Hypothesis** can be included in the daily bulletin. Each **Event Hypothesis** may also have a relationship to any number of **Event Screening Category** objects (also described in Section 2.6.9).

2.6.6 Location Solution

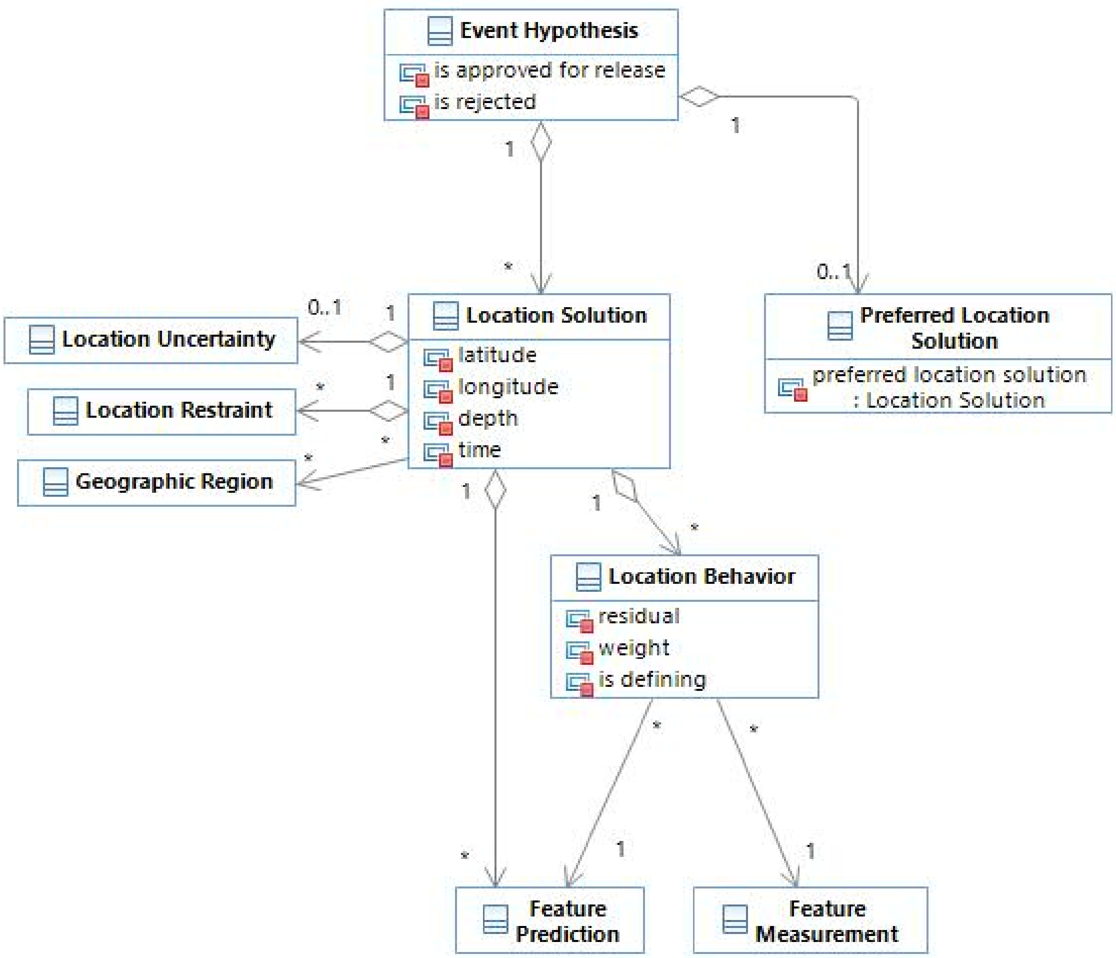

Figure 17. Location Solution Classes

A **Location Solution** (Figure 17) includes the attributes **latitude**, **longitude**, **depth**, and **time.** A **Location Solution** is often determined by a location algorithm that minimizes the difference between **Feature Measurements** (usually arrival time, azimuth, and slowness) and corresponding **Feature Predictions**. Both of these are linked to the **Location Solution** through the **Location Behavior** class. **Location Behavior** includes an attribute (**is defining**) to indicate whether this measurement is used in the location. Note that **Feature Predictions** may also be directly associated to a **Location Solution** to support predictions where no measurement is available (e.g. a non-detecting station).

For the same **Event Hypothesis** (i.e., the same set of associated **Signal Detection Hypotheses**), multiple **Location Solution**s can be formed by using different location constraints (represented in the **Location Restraint** class) such as depth (i.e. unconstrained, fixed to surface, fixed to a depth below the surface), location, or time. One reason for having multiple solutions is that comparing how well the **Feature Measurement**s fit with an unconstrained **Location Solution** versus a constrained **Location Solution** is a reliable method to assess how reasonable the constraints are. Results from these types of comparisons facilitate event screening based on source type. For instance, when an event's depth is determined to be both well resolved and appreciably below the Earth's surface, it can be screened out as non-nuclear. Multiple **Location Solutions** can also be formed by using multiple location algorithms (e.g., Geiger's method, grid search). For each **Event Hypothesis** with multiple **Location Solution**s, one **Location Solution** must be designated as the **Preferred Location Solution**.

Each **Location Solution** may also have one or more Geographic Regions associated with it. This allows grouping of events originating within the same geographic bounds (i.e., continent, country, or arbitrary polygon). A **Location Solution** may also have a **Location Uncertainty** associated with it, although it is not required. The details of **Location Uncertainty** are outlined in the next section.

2.6.7 Location Uncertainty

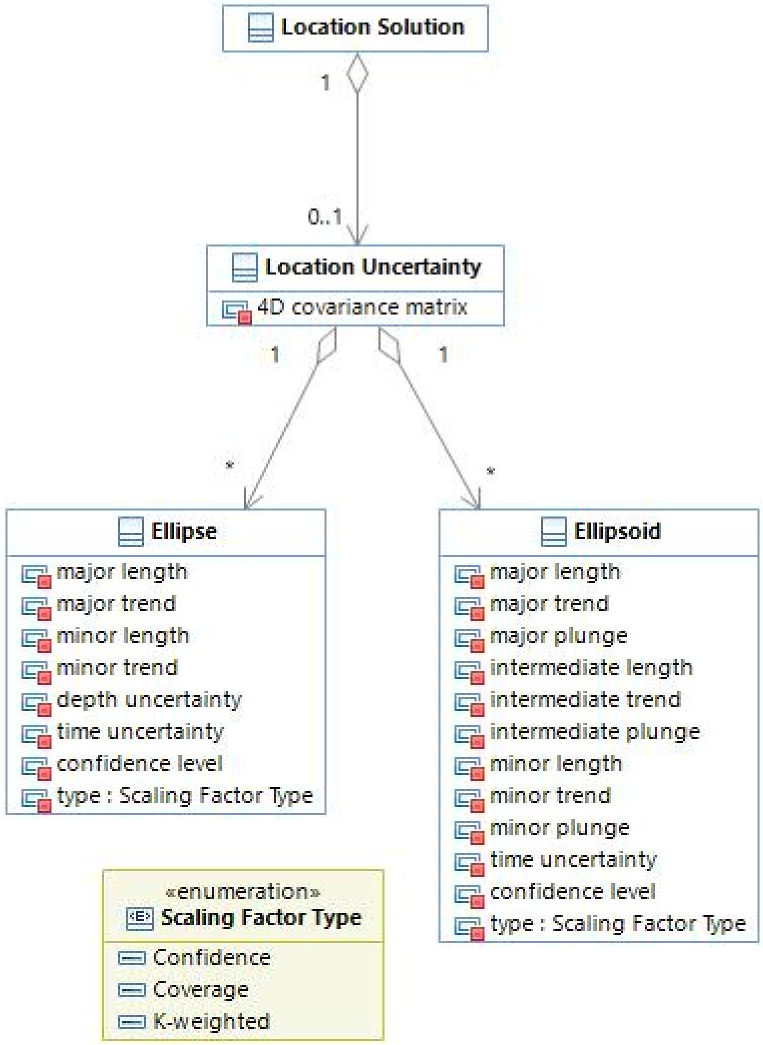

Figure 18. Location Uncertainty Classes

Each **Location Solution** can have a **Location Uncertainty** (Figure 18). This is optional to allow cases where it is not available (e.g., if an event catalog is imported that does not have this information). **Location Uncertainty** information spans four dimensions (time, latitude, longitude, depth), which are captured in the **4D Covariance Matrix** attribute. However, for many purposes, it is preferable to calculate a projection of the covariance matrix as a 2D **Ellipse** (in latitude and longitude) or 3D **Ellipsoid** (in latitude, longitude, and depth). Both of these projection classes include various attributes that describe the orientation of their basic geometric shape, as well as uncertainty for additional dimensions not included in that shape.

Note that the **Location Uncertainty** class described here assumes that a linearized location method has been used. If a non-linear method is used, this formulation cannot fully capture the uncertainty information and a new set of classes may be needed.

2.6.8 Magnitude Solutions

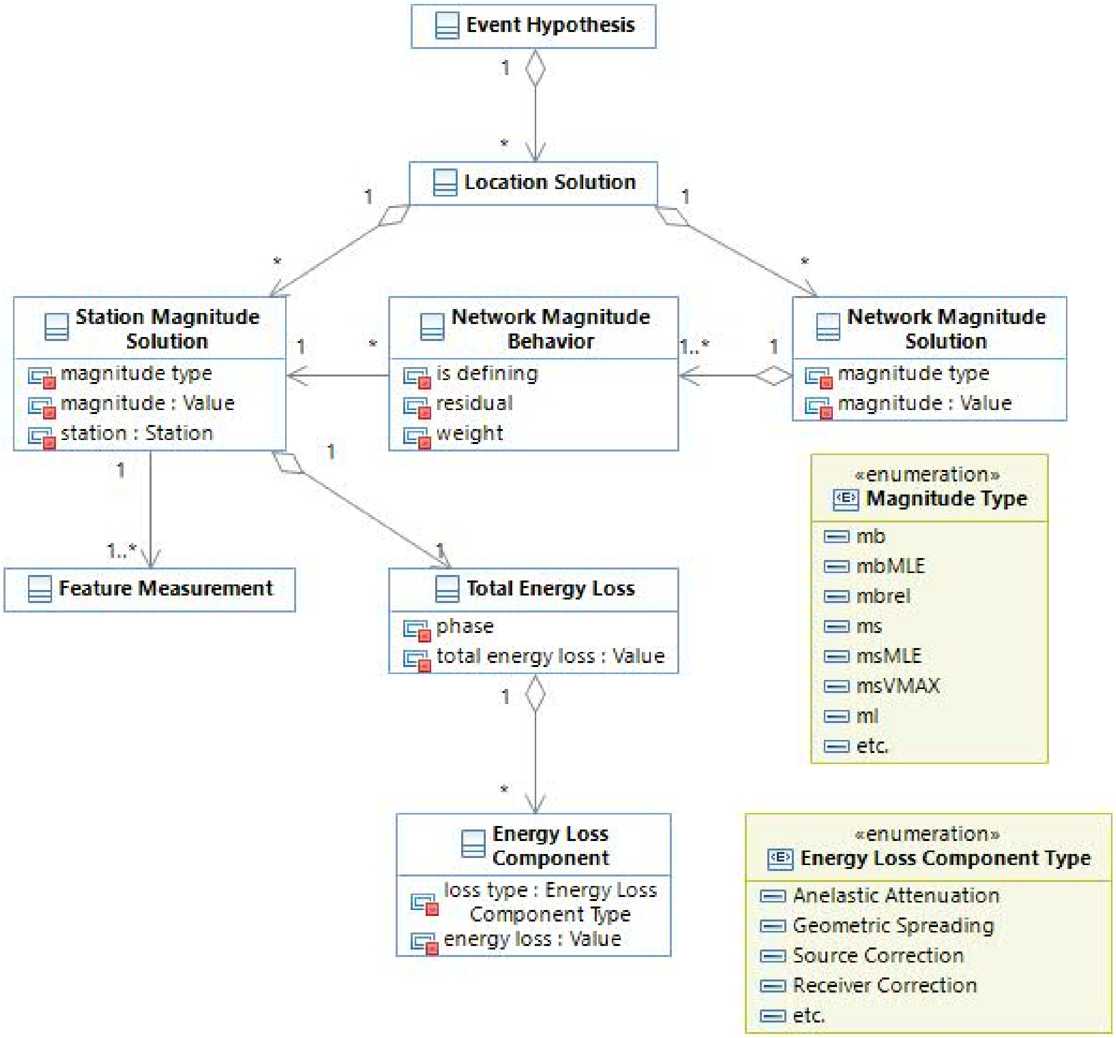

Figure 19. Magnitude Solutions Classes

Magnitude is a measure of the size of an **Event** (Figure 19). Each **Event Hypothesis** can have multiple magnitude estimates of different types (see Magnitude Type enumeration list). Magnitude estimates are dependent upon a **Location Solution**. In this model, a **Location Solution** aggregates all **Magnitude Solutions** calculated using it; this includes **Station Magnitude Solutions** that directly use the **Location Solution** in their calculations, and **Network Magnitude Solutions** that are calculated from those **Station Magnitude Solutions**.

A **Station Magnitude Solution** is dependent on one or more **Feature Measurements** (e.g., amplitude and period), a **Location Solution**, and **Total Energy Loss**. The **Total Energy Loss** is a path-dependent prediction that requires knowing both the **Station** location and **Event Hypothesis** location (which is in the **Location Solution**). It is an aggregate of different **Energy Loss Components** (e.g., anaelastic attenuation, geometric spreading, source and receiver corrections).

A **Network Magnitude Solution** is dependent on a collection of **Station Magnitude Solutions**, each of which must be made for the same **Location Solution**. The **Station Magnitudes** that are grouped with the **Network Magnitude Solution** are indicated in the **Network Magnitude Behavior** class, which includes an attribute (**is defining**) to indicate whether a particular **Station Magnitude Solution** is used in the **Network Magnitude Solution** calculation.

2.6.9 Event Screening & Bulletin Inclusion

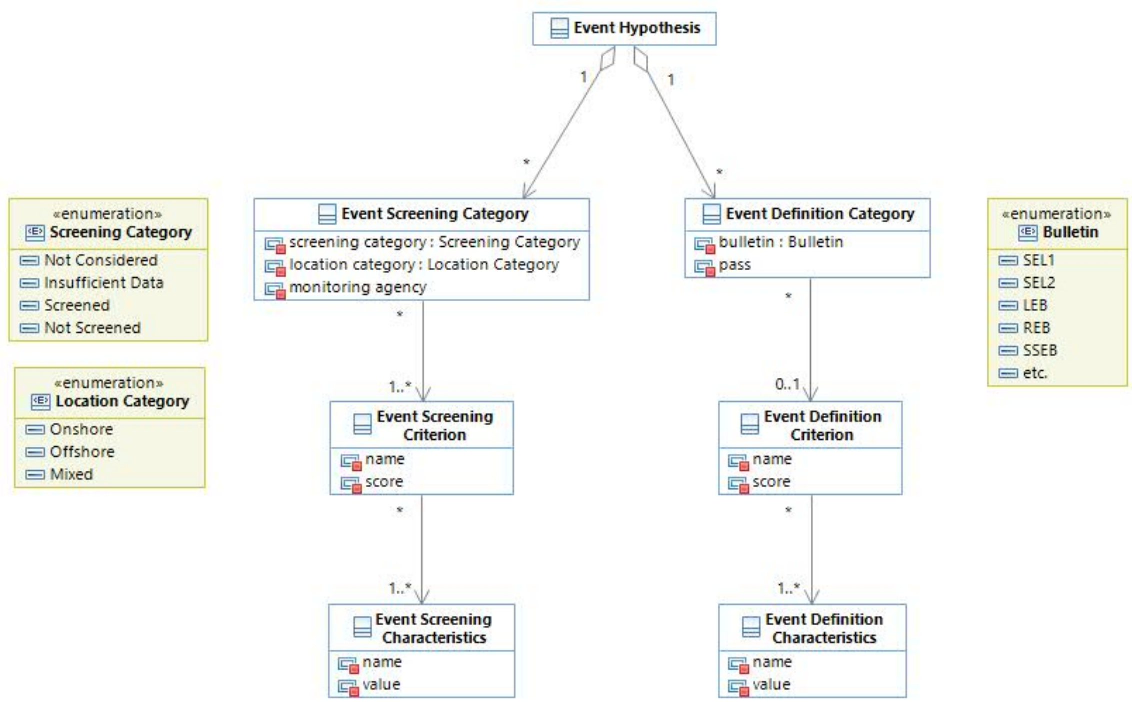

Figure 20. Event Screening & Bulletin Inclusion Classes

The IDC has minimum standards, known as **Event Definition Criteria** (Figure 20, right hand side), for the amount and type of information required for events to be included in their bulletins (e.g., the event must be associated to three or more primary network stations). **Event Definition Characteristics** are computed for each **Event Hypothesis** and are used to calculate scores for one or more **Event Definition Criterion**. The **Event Definition Criterion** scores determine if the **Event Hypothesis** is included in a given bulletin. This information is summarized in the **Event Definition Category** class, where the relevant **bulletin** and inclusion determination (**pass**) are indicated.

For some bulletins, event screening is applied to attempt to remove certain types of events, e.g. any obviously natural events if the bulletin is only supposed to include possible explosions. This is also shown in Figure 20 (left hand side). **Event Screening Characteristics** (mb, Ms, various regional phase amplitudes, etc.) are computed for each **Event Hypothesis** that is to be screened**.** The **Event Screening Characteristics** are used to calculate scores for the **Event Screening Criterion** (e.g. Ms:mb, regional phase ratios, etc.). Based on the scores for several

Event Screening Criteria, each **Event Hypothesis** is assigned to one or more **Event Screening Categories**. The **Event Screening Category** class includes both a **screening category** and a **location category**, both of which are enumerated types as shown in the figure above. The **monitoring agency** can be the IDC or any member state, in the case where they choose to set up their own custom screening (i.e. using different **Event Screening Characteristics** and/or different score thresholds for the **Event Screening Criterion**.

3 Examples

3.1 Three-component Filter and Signal Detection Configuration

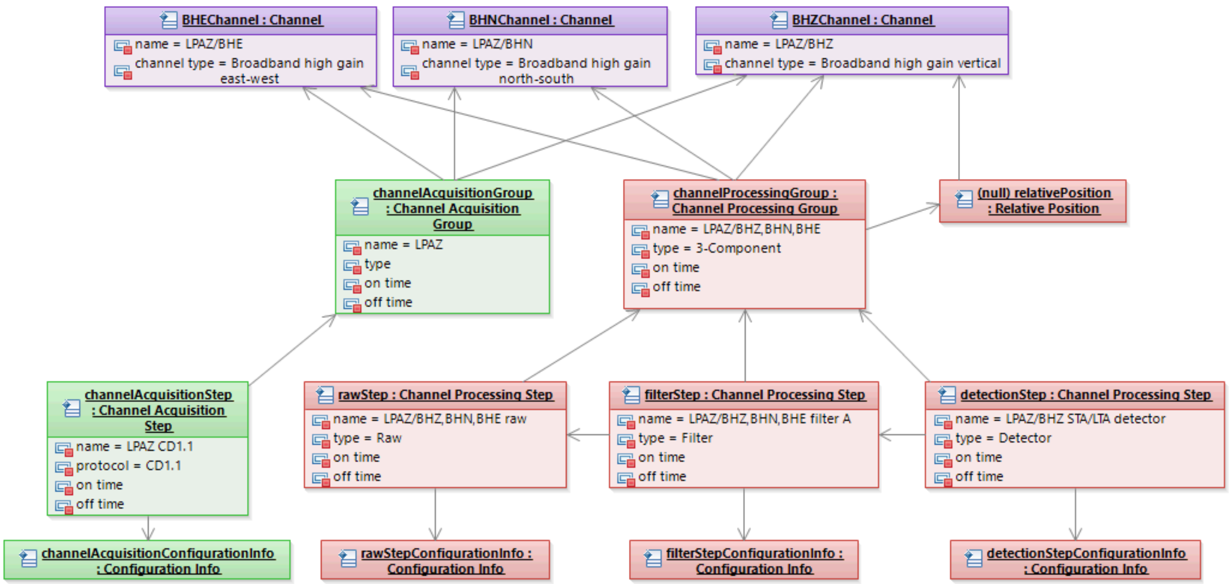

Figure 21. Example: Three-component Filter and Signal Detection Configuration

This example (Figure 21) shows the classes involved in configuring filtering and single-channel (BHZ) signal detection on a three-component **Station** (LPAZ). We begin with the three **Channels** (BHZ, BHN, BHE) that produce the waveform data for this station. To bring data from these **Channels** onto the System, we first have to assign them to a **Channel Acquisition Group**, which has an **on time** and an **off time**, for which the **Channel Acquisition Group** is valid. In this case, we are grouping multiple **Channels** from the same **Station**; hence, the **name** of the **Channel Acquisition Group** is "LPAZ," but the grouping is intended to be fully general, such that **Channels** from different **Stations** can be grouped, as well. Next, we assign a **Channel Acquisition Step** to the **Channel Acquisition Group**, which specifies how data from the **Channels** in the **Channel Acquisition Group** will be acquired. The **Channel Acquisition Step** includes the **protocol** that will be used for transmitting the data, and it links to **Configuration Info,** which provides details about the acquisition.

Once the data are available on the System, we can process them for explosion monitoring purposes. The first step is making the raw data available. The data have been acquired in the **Channel Acquisition Step** described above and exist somewhere on the Data Acquisition Partition, but they must actually be read into memory for data processing. We begin by setting up a **Channel Processing Group** ("LPAZ/BHZ, BHN, BHE") that includes all three **Channel** members. The actual detection processing will only involve data from the BHZ **Channel**, but setting up a **Channel Processing Group** with all three **Channels** as members allows us to reuse that same **Channel Processing Group** for other purposes, which may involve the other channels (e.g., polarization analysis). The **Relative Position** class captures the position of each **Channel** within the **Channel Processing Group** relative to a reference location (which is captured with a set of attributes within the **Channel Processing Group**). However, in this case, all the **Channels** are sited at the same location, so **Relative Position** is irrelevant, and we show these relationships as null. To this **Channel Processing Group**, we now assign a **Channel Processing Step**, in this case "LPAZ/BHZ, BHN, BHE raw," which just loads the raw waveform data from the data acquisition partition, and makes them available. This **Channel Processing Step** has **Configuration Info** associated with it that describes the details of the data loading.

With the data available, we can now apply a filter. Again, we use the same **Channel Processing Group**. We link to yet another **Channel Processing Step**, in this case "LPAZ/BHZ, BHN, BHE filter A," and it, in turn, points to **Configuration Info** that provides detailed information about how the filtering is to be done. Note that in this example this step will filter data for all three **Channels**, though only one of these will be used in the signal detection step that comes next. Note also that this **Channel Processing Step** points back to the **Channel Processing Step** that made the raw waveform data available ("LPAZ/BHZ, BHN, BHE raw"), as filtering is intended to be run on the raw data.

Finally, we can configure the signal detection step, which is run on the filtered data from the BHZ channel. Again, we use the same **Channel Processing Group**, though the processing will just make use of data from the BHZ **Channel**. We assign a new **Channel Processing Step** "LPAZ/BHZ STA/LTA Detector," which points to **Configuration Info** that provides details about how the STA/LTA detection is done, and to the filter **Channel Processing Step** that provided the filtered waveform data that the detector is running on.

3.2 Three-component Filter and Signal Detection Configuration with Results

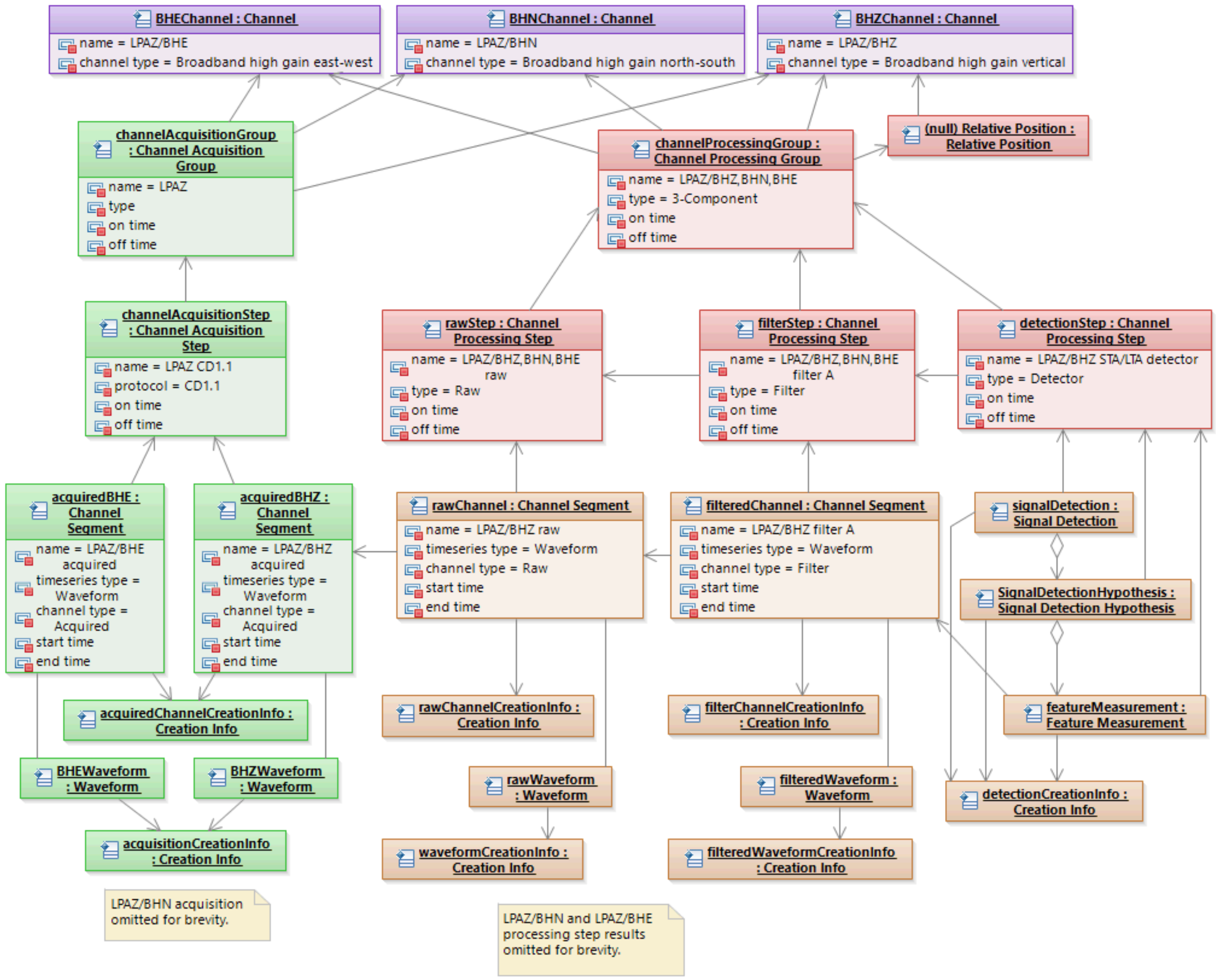

Figure 22. Example: Three-component Filter and Signal Detection Configuration with Results

In this example (Figure 22), we extend the diagram from the previous configuration example, to show classes that are used when we actually begin to acquire and process data to produce signal detections on LPAZ/BHZ. Once the System begins to acquire data, then the **Channel Acquisition Step** (as discussed in the previous example) produces three **Channel Segments**, one for each of the **Channels**, though for brevity we only show two: “LPA/BHZ acquired” and “LPAZ/BHE acquired”. Note that each **Channel Segment** represents a specific time interval of data (bracketed with **start time** and **end time**) from one of the **Channels**. Every time a new interval of data is acquired for either of the **Channels,** a new **Channel Segment** object will be created. Each **Channel Segment** points to **Creation Info** that captures details about how the segment of data represented by the **Channel Segment** was acquired. Each of the **Channel Segment** objects also points to a **Waveform** that contains the actual data segment that the **Channel Segment** record describes. Each of these **Waveforms** also points to **Creation Info** that captures details about how the segment of data was created.

Moving to the right side of the figure, where the data processing is covered, we see similar extensions to the previous example. The **Channel Processing Step** ("LPAZ/BHZ, BHN, BHE raw") that made the raw waveform data available for processing, now points to **Channel Segments** for every raw data interval that will be processed, though in the example we only show one for the BHZ **Channel** ("LPAZ/BHZ raw"), for brevity. As with the acquisition, this **Channel Segment** object points to a **Waveform** object, and both point to **Creation Info** that captures details about how each segment of raw data was made available for processing.

The filter **Channel Processing Step** does produce new time series data for all three **Channels**, though our example just shows what happens for BHZ. We show a new **Channel Segment** ("LPAZ/BHZ filter A") as well as a corresponding **Waveform**. Both point to **Creation Info** that captures the details of how those data segments were created. The filter **Channel Segment** also points to the raw **Channel Segment** from which it was derived.

The last processing operation is the signal detection **Channel Processing Step** ("LPAZ/BHZ STA/LTA detector"). This step does not produce a time series, so there is no **Channel Segment** or **Waveform**. However, it does produce a **Signal Detection**, a **Signal Detection Hypothesis** (this has to be created when a **Signal Detection** is created), and a **Feature Measurement** (every **Signal Detection Hypothesis** must have at least an arrival time measurement). All three of these objects point to **Creation Info** that provides details about how they were created. Only the **Feature Measurement** is actually made on the time series data and hence points to the filter **Channel Segment** object.

3.3 *Array Filter, Beam, and Detection Configuration*

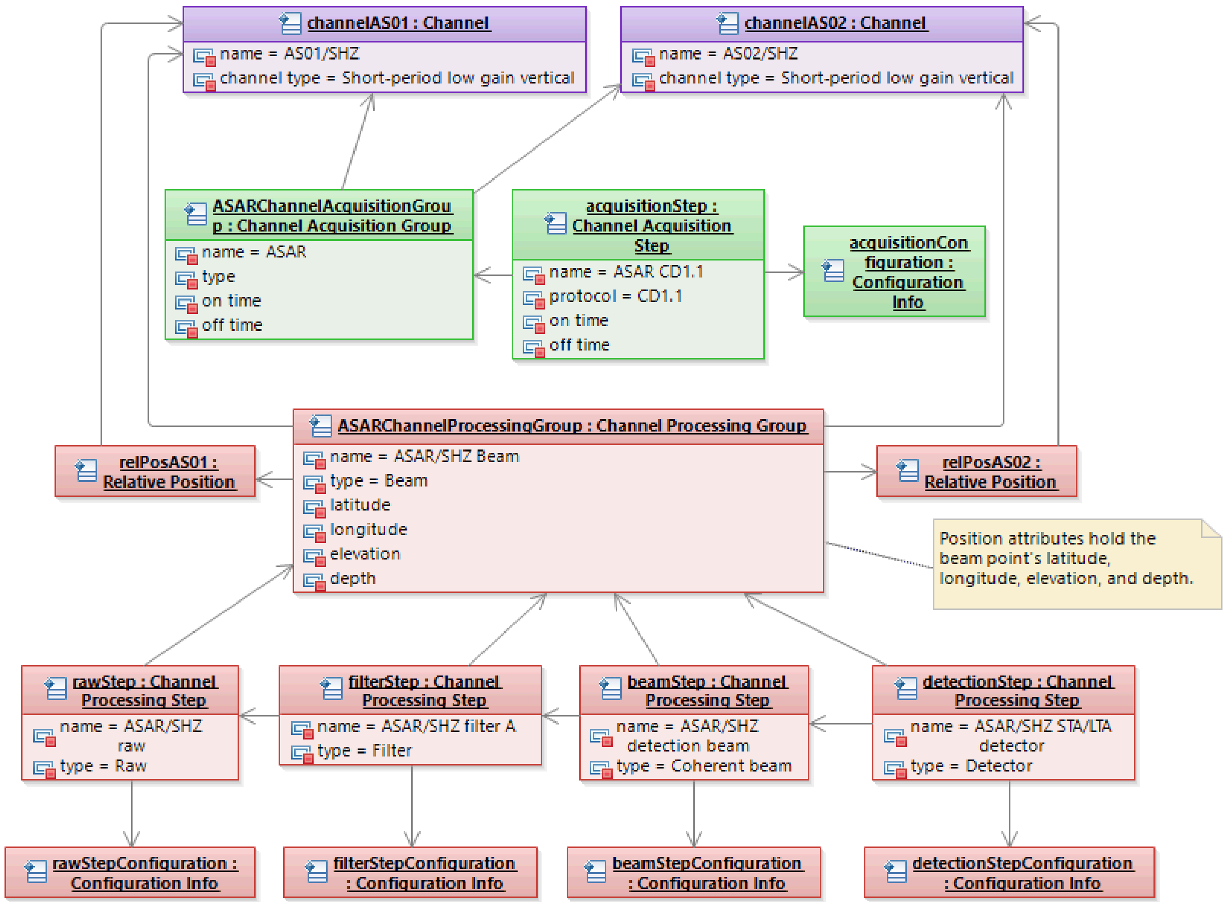

Figure 23. Example: Array Filter, Beam, and Detection Configuration.

This example (Figure 23) shows the classes involved in configuring filtering, beaming, and signal detection for an array **Station** (ASAR). To limit the complexity of our diagram, our ASAR array only has two **Channels** (AS01/SHZ and AS02/SHZ). The acquisition configuration is very similar to the three-component **Station** example. We begin with the **Channels** that produce the waveform data. To bring data from these **Channels** onto the System, we first have to assign them to a **Channel Acquisition Group** ("ASAR"), which has an **on time** and an **off time**, for which the **Channel Acquisition Group** is valid. Next, we assign a **Channel Acquisition Step** to the **Channel Acquisition Group**, which specifies how data from the **Channels** in the **Channel Acquisition Group** will be acquired. The **Channel Acquisition Step** includes the **protocol** that will be used for transmitting the data, and it also links to **Configuration Info** that provides details about the acquisition.

Once the data are available on the System, we can process them for explosion monitoring purposes. The first step is making the raw data available. We begin by setting up a **Channel Processing Group** ("ASAR/SHZ Beam") that includes both **Channel** members. In this array example, the location information within the **Channel Processing Group** (**latitude**, **longitude**,

elevation, **depth**) defines the beam point, and hence needs to be specified. Further, the **Relative Position** class captures the relative position of each **Channel** with respect to that location; hence, these classes are not null. To this **Channel Processing Group**, we now assign a **Channel Processing Step**, in this case “ASAR/SHZ raw,” which just loads the raw waveform data from the data acquisition partition and makes them available. This **Channel Processing Step** has **Configuration Info** associated with it that describes the details of the data loading.

With the data available, we can apply a filter. Again, we use the same **Channel Processing Group**. We link to yet another **Channel Processing Step**, in this case “ASAR/SHZ filter A,” and it in turn points to **Configuration Info** that provides detailed information about how the filtering is to be done. Note that this **Channel Processing Step** points back to the **Channel Processing Step** that made the raw waveform data available (“ASAR/SHZ raw”), as filtering is intended to be run on the raw data.

For this example, we have an additional waveform processing step to form the beam. Again, we use the same **Channel Processing Group**. We link to another **Channel Processing Step**, in this case “ASAR/SHZ detection beam,” and it, in turn, points to **Configuration Info** that provides detailed information about how the beaming is to be done, using the beam point that was specified in the **Channel Processing Group**. Note that this **Channel Processing Step** points back to the **Channel Processing Step** that made the filtered waveform data available (“ASAR/SHZ filter A”), as beaming is intended to be run on the filtered data.

Finally, we can configure the signal detection step, which is run on the beamed data from the two SHZ **Channels**. Again, we use the same **Channel Processing Group**. We assign a new **Channel Processing Step** “ASAR/SHZ STA/LTA Detector”, which points to **Configuration Info** that provides details about how the STA/LTA detection is done, and back to the beam **Channel Processing Step** that provided the beam waveform data on which the detector is running.

3.4 *Array Filter, Beam, and Detection Configuration with Results*

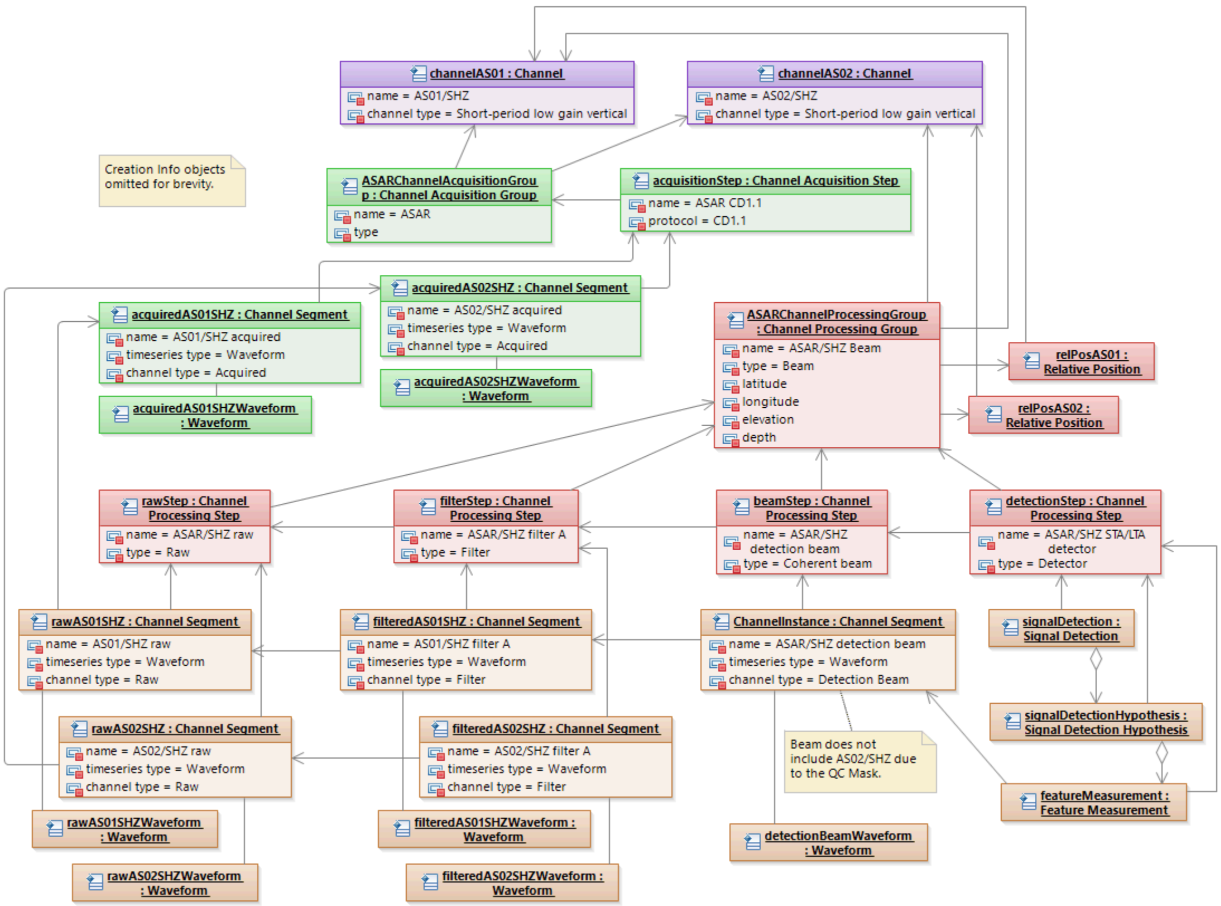

Figure 24. Example: Array Filter, Beam, and Detection Configuration with Results

We extend the diagram from the previous array configuration example to show classes that are used when we actually begin to acquire and process data to produce signal detections on an ASAR/SHZ detection beam (Figure 24). Once the System begins to acquire data, then the **Channel Acquisition Step** (as discussed in the previous example) produces two **Channel Segments**, one for each of the **Channels** ("AS01/SHZ acquired" and "AS02/SHZ acquired"). As described in the three-component example, each of these **Channel Segments** represents a specific time interval of data from one of the **Channels**. Each **Channel Segment** points to **Creation Info** that captures details about how that segment of data represented by the **Channel Segment** was acquired. Each of the **Channel Segment** objects also points to a **Waveform** that contains the actual data segment the **Channel Segment** record describes. Each of these **Waveforms** also points to **Creation Info** that captures details about how the segment of data was created.

Moving to the right side of the figure, where the data processing steps are covered, we see similar extensions to the previous array configuration example. The **Channel Processing Step** ("ASAR/SHZ raw"), which made the raw waveform data available for processing, now points to

Channel Segments for every raw data interval that will be processed. As with the acquisition, each of these **Channel Segment** objects points to a **Waveform** object, and both point to **Creation Info** that captures details about how each segment of raw data was made available for processing, though we do not show that relationship on the diagram, to limit complexity.

The filter **Channel Processing Step** produce new time series data for both **Channels**. We show new **Channel Segments** (“AS01/SHZ filter A,” “AS01/SHZ filter A”) as well as a corresponding **Waveform** for each. Both point to **Creation Info** that captures the details of how those filtered data segments were created. Each of the filtered **Channel Segments** also points to the raw **Channel Segment** from which it was derived.

Next comes the beam **Channel Processing Step**, which produces a single stream of time series data; hence, we show a single new **Channel Segment** (“ASAR/SHZ detection beam”). This **Channel Segment** points to a corresponding **Waveform**, and both objects point to **Configuration Info** that provides detailed information about how the beaming was done. Note that in this example, there was a QC problem with the AS02/SHZ data for the interval corresponding to the **Channel Segment**; hence, the beam does not include AS01/SHZ data. This is why the beam **Channel Segment** only points to the filtered **Channel Segment** for AS01/SHZ instead of to both **Channel Segments**. This illustrates an important feature of our data model. The configuration information in the **Channel Processing Steps** records how the System was supposed to do the processing, whereas the linkage of the **Channel Segment** information through the processing steps shows what actually happened.

The last processing operation is the signal detection **Channel Processing Step** (“ASAR/SHZ STA/LTA detector”). This step does not produce a time series, so there is neither a **Channel Segment** nor a **Waveform**. However, it does produce a **Signal Detection**, a **Signal Detection Hypothesis** (this has to be created when a **Signal Detection** is created), and a **Feature Measurement** (every **Signal Detection Hypothesis** must have at least an arrival time measurement). All three of these objects point to **Creation Info** that provides details about how they were created. Only the **Feature Measurement** is actually made on the time series data and hence points to the filter **Channel Segment** object.

3.5 *Three-component QC Mask Configuration with Results*

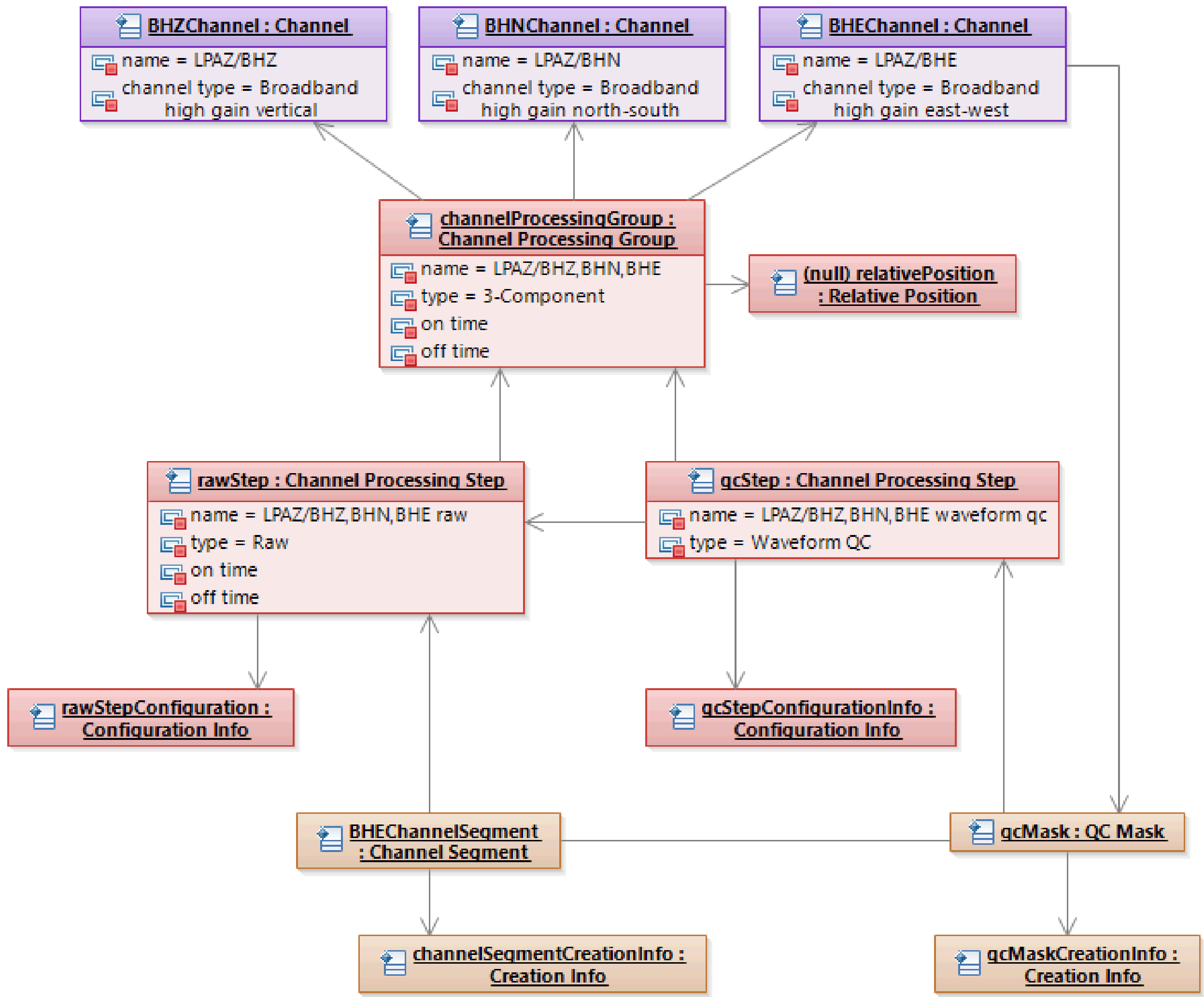

Figure 25. Example: Three-component QC Mask Configuration with Results

In this example (Figure 25), we show the classes that are used when we perform QC Masking on all three **Channels** for station LPAZ. A more complicated example of data processing for LPAZ was shown in Figure 22, so we only include the important parts for QC Masking here. We omit the data acquisition steps (shown on the left side of Figure 22) and begin with the data processing. The **Channel Processing Step** ("LPAZ/BHZ, BHN, BHE raw") makes the raw waveform data available for processing, and points to **Channel Segments** for every raw data interval that will be processed, though in the example we only show one for the BHE **Channel** ("BHEChannelSegment"), for brevity. **Creation Info** captures details about how each segment of raw data was made available for processing.

The QC **Channel Processing Step** ("LPAZ/BHZ,BHN,BHE waveform qc") does not produce new time series data for any of the **Channels**, hence no **Channel Segment**, but it does produce a **QC Mask**, which points back to the Channel Segment that it was calculated for, as well as to **Creation Info** that captures details about how the masking.

3.6 *Network Processing Configuration*

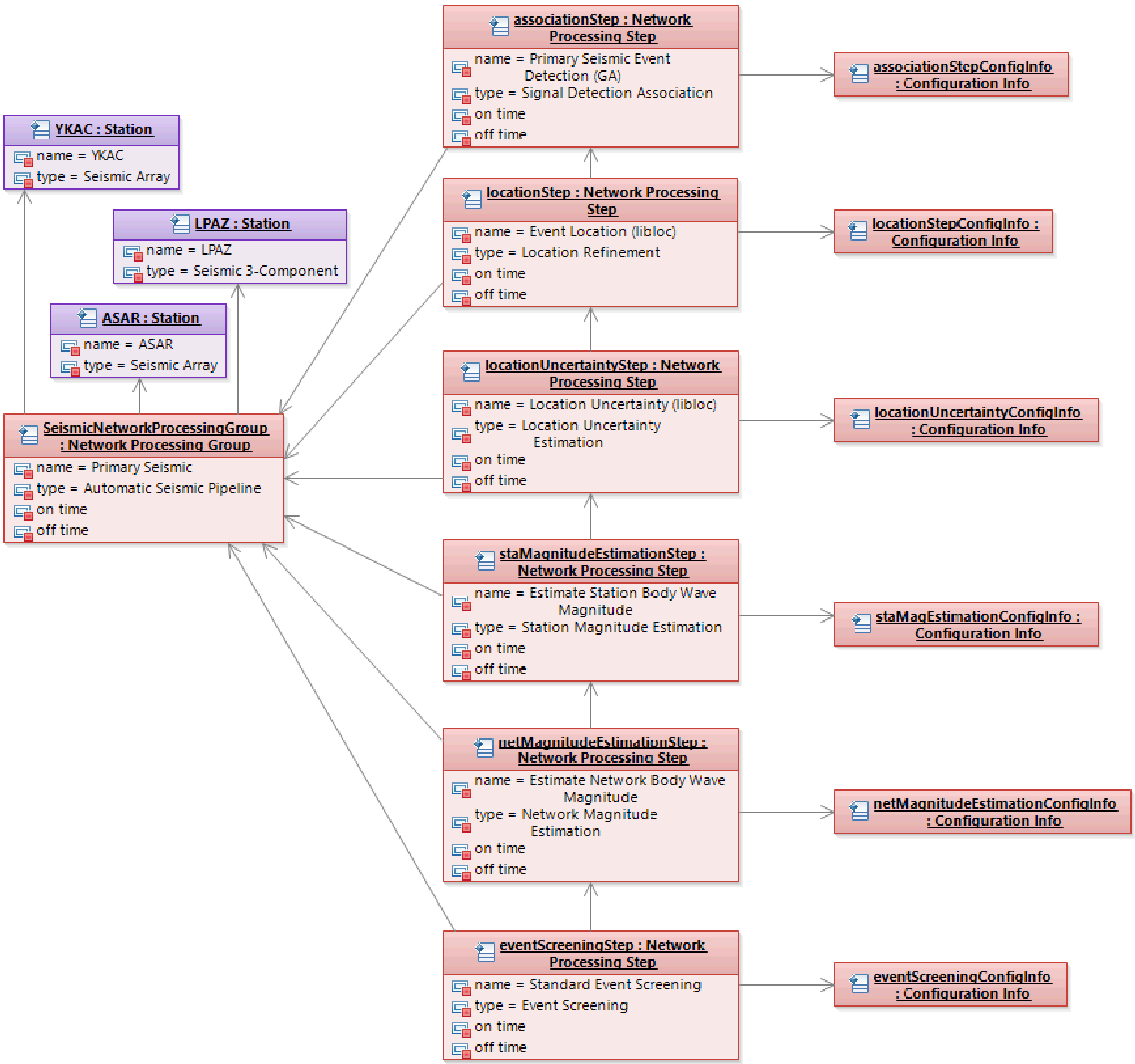

Figure 26. Example: Network Processing Configuration

In this example (Figure 25), we show the classes involved in configuring a series of processing steps in a network processing pipeline. We being with a **Network Processing Group** ("Primary Seismic"), that groups together data from a specified set of stations (YKAC, LPAZ, ASAR). All of the **Network Processing Steps** in our pipeline will refer back to this same **Network Processing Group** and hence are limited to using the stations in this group**,** though each step can make its own decision about which of the stations to use. Our first **Network Processing Step** builds events out of signal detections and has type "Signal Detection Association" and name "Primary Seismic Event Detection (GA)". As with all steps, this points to **Configuration Info**, which provides details about the software and parameters used for this step.

The next **Network Processing Step** (type "Location Refinement", name "Event Location (libloc)",) refines the locations of the events built in the previous step, which it references as a parent **Network Processing Step**. Again, there is associated **Configuration Info**. This step is followed by more **Network Processing Steps** to estimate location uncertainty (type "Location Uncertainty Estimation", name "Location Uncertainty (libloc)"), to estimate body wave magnitudes for each station (type "Station Magnitude Estimation", name "Estimate Station Body Wave Magnitude"), to estimate body wave magnitude across the network (type "Network Magnitude Estimation", name "Estimate Network Body Wave Magnitude"), and finally to screen out obvious non-explosions (type "Event Screening", name "Standard Event Screening"). Note that each of these refers back to the previous step as a parent **Network Processing Step**.

3.7 Network Processing Configuration with Results

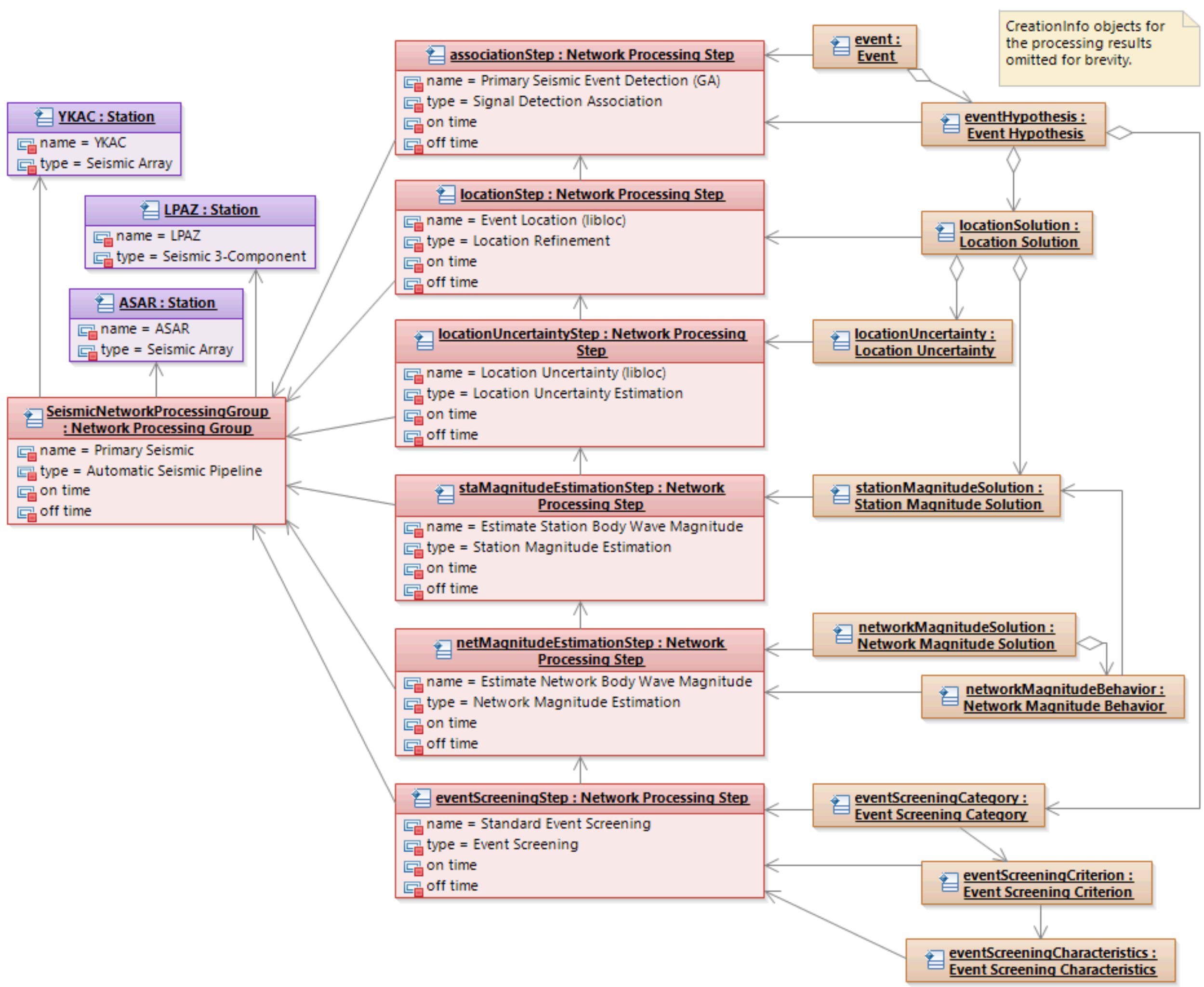

Figure 27. Example: Network Processing Configuration with Results

We extend the diagram from the network processing configuration example to show classes that are used when we actually begin to process data to build and characterize events (Figure 27). The left hand side of the diagram is the same as before, but on the right hand side we have omitted the **Configuration Info** that is linked to each **Network Processing Step** and instead shown the various data product classes that are produced by each step, and that were discussed in detail previously in this document. The “Event Location” step produces **Events** and **Event Hypotheses** (each **Event** must have at least one **Event Hypothesis**). This step would also create the **Association** classes that link each **Event Hypothesis** to the set of **Signal Detection Hypotheses** it includes, but we do not show these here to keep the diagram from getting too cluttered.

The “Location Refinement” step will create one or more **Location Solutions** for each **Event Hypothesis** from the previous step (depending on how this step is configured). Next, the “Location Uncertainty Estimation” step creates one or more **Location Uncertainties** for each of the **Location Solutions** from the previous step.

With robust event locations, we can estimate magnitudes, hence the next step is “Station Magnitude Estimation”, which calculates **Station Magnitude Solutions** for all of the stations within the **Network Processing Group** that are associated with each **Event Hypothesis**. Note that each of these is aggregated by a particular **Location Solution** (station magnitudes cannot be estimated without an event location), which we show, and they also reference a **Signal Detection Feature Measurement**, which we do not show here.

The “Network Magnitude Estimation” step calculates a **Network Magnitude Solution** estimate based on one or more **Station Magnitude Solutions.** The subset of the full set of Station Magnitude Solutions that are actually used to calculate the Network Magnitude (i.e. the “defining” stations) are captured in the **Network Magnitude Behavior** class.

Finally, the “Event Screening” step determines an **Event Screening Category** for each **Event Hypothesis**, which is based on one or more **Event Screening Criteria** (e.g. Ms:mb, Regional P/S), which are in turn based on one or more **Event Screening Characteristics**. (Ms, mb, regional amplitudes, etc.).

4 References

Ahern, T., R. Styles, K. Skjellerup, R. Casey, D. Barnes, R. Benson, T. Knight, C. Trabant, 2012, SEED Reference Manual Version 2.4, Incorporated Research Institutions for Seismology, Seattle, Washington.

Anderson, J., W.E. Farrell, K. Garcia, J. Given, H. Swanger, 1990, Center for Seismic Studies Version 3 Database Schema Reference Manual, Technical Report C90-01 SAIC-90/1235, 64pp, Science Applications International Corporation, San Diego, California.

Schorlemmer, D., J. Saul, F. Euchner, J. Becker, R. Buland, A. Heinloo, L. Kamb, P. Kastli, A. Spinuso, B. Weber, 2013, QuakeML – An XML Representation of Seismological Data Basic Event Description Version 1.2.